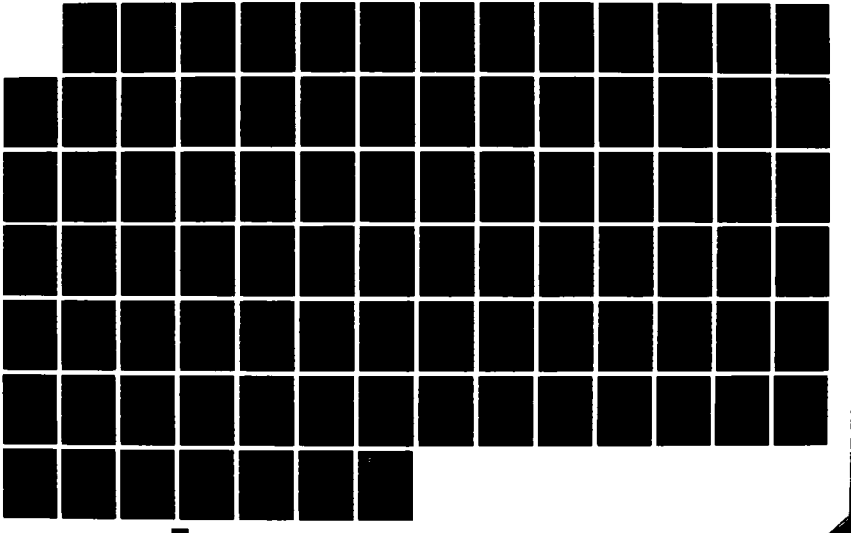
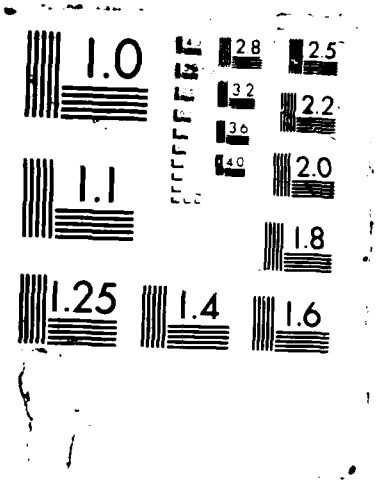
AD-A190 140 EXHAUSTIVE AND PSEUDO-EXHAUSTIVE TESTING(U) STANFORD UNIV CA CENTER FOR RELIABLE COMPUTING E J MCCLUSKEY JUL 87 CRC-TR-87-13 N00014-85-K-0600 1/1

UNCLASSIFIED F/G 12/6 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

Center for Reliable Computing

Exhaustive and Pseudo-Exhaustive Testing

by

E.J. McCluskey

CRC Technical Report No. 87-13

(CSL TN No. 87-331)

July 1987

CENTER FOR RELIABLE COMPUTING
Computer Systems Laboratory
Electrical Engineering and Computer Science Departments
Stanford University, Stanford CA 94305-4055

This work was supported in part by the Innovative Science and Technology Office of the Strategic Defence Initiative Organization administered through the Office of Naval Research under contract No. N00014-85-K-0600.

Exhaustive and Pseudo-Exhaustive Testing

by

E.J. McCluskey

CRC Technical Report No. 87-13

(CSL TN No. 87-331)

July 1987

CENTER FOR RELIABLE COMPUTING
Computer Systems Laboratory
Electrical Engineering and Computer Science Departments
Stanford University, Stanford CA

ABSTRACT

This Technical Report Contains a tutorial to be presented at the International Test Conference, Washington, D.C., Sept. 1-3, 1987.

Table of Contents

EXHAUSTIVE AND PSEUDO-EXHAUSTIVE TESTING

E.J. McCluskey

CENTER FOR RELIABLE COMPUTING
Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University, Stanford, CA 94305-4055

INTRODUCTION

This chapter discusses an approach to test pattern generation for functional testing of combinational digital integrated circuits. The techniques to be presented apply all possible input patterns either to the entire circuit under test or to portions of the circuit under test.

For some circuits, exhaustive testing is economically practical: all possible input combinations are applied to the entire circuit. An exhaustive test is a complete functional test and no assumptions about the internal structure of the circuit under test are required. When the number of inputs is so large that an exhaustive test is impractical, it is still possible to retain many of the advantages of exhaustive testing by using a pseudo-exhaustive test technique. Pseudo-exhaustive testing requires that some details of the internal circuit structure either be known or assumed.

The minimum structural information required is the knowledge of which inputs affect each of the outputs. This information permits each of the circuit outputs to be tested exhaustively, a pseudo-exhaustive test called output function verification. If output function verification requires too long a test, additional structural information is required in order to permit segmenting the circuit so that each segment can be tested exhaustively, a technique called segment verification or segment testing. The amount of structure that must be known depends on the circuit and the acceptable test length.

Exhaustive and pseudo-exhaustive testing guarantee 100% single-stuck fault coverage. The coverage of other classes of faults is discussed in connection with each specific technique.

A disadvantage of exhaustive and pseudo-exhaustive testing is the size of the test sets: they can be substantially larger than minimum-length test sets. Test set size affects the amount of memory to store the set and the time to apply the test. For exhaustive or pseudo-exhaustive testing, the test application time can be long but the memory required to store the test is very small due to the regularity of the test sets.

The major advantages of exhaustive and pseudo-exhaustive tests are the extremely high fault coverage, the generality of the fault model, and the absence of a requirement to do fault simulation. Another advantage of output function verification is the ease of generating the test set. Segmentation and determination of the corresponding test sets requires more computation.

BOOLEAN TESTING

The words test and testing as used here refer only to "Boolean Testing" of digital integrated circuits. A Boolean Test is defined as a test in which only logical signal values are considered. Parametric tests of digital circuits and testing analog circuits are specifically excluded from the discussion. Verifying the correctness of a design also differs from Boolean testing. They are closely related since the same vectors are sometimes used for both purposes.

Definition: Design verification establishes that a design correctly implements a behavioral specification. The customary technique for design verification is a fault-free simulation of the circuit operation. This simulation uses a set of "design verification" or "simulation" vectors as inputs. The design verification vectors are typically generated manually by the designer. If the correct output values are present for each of the design verification inputs, it is assumed that the design is correct.

Table 1 shows design verification vectors for the dual 4-line to 1-line multiplexer of Fig. 1. Part a of the table lists a set of 8 vectors obtained from [Franz 85]. These vectors are derived by assuming that it is sufficient to verify that both a 0- or a 1-signal on the addressed input are passed correctly to the output. Part b lists 20 vectors contributed by Prof. Jan Hlavicka. He assumed that it was necessary to make sure that a 1-signal on a nonaddressed line wasn't propagated to the circuit output.

It's not surprising that different approaches lead to different design verification vectors since there isn't any standard approach to generating such vectors. In fact, the only way to truly guarantee the correctness of a design would be:

> (1) to have a complete specification of the desired output for every possible input, including an identification of those inputs for which the output was not specified, and
>
> (2) to do a detailed simulation of the circuit response to all possible inputs.

It is usually not considered feasible to do such a thorough verification of a design. Instead, the designer makes some assumptions about the implementation and selects design verification vectors based on those assumptions.

Design verification vectors are discussed here only because they are often used for Boolean testing as well as design verification. Before discussing this technique, a more precise definition of Boolean testing will be given. The Fig. 1 multiplexer will be used as a running example.

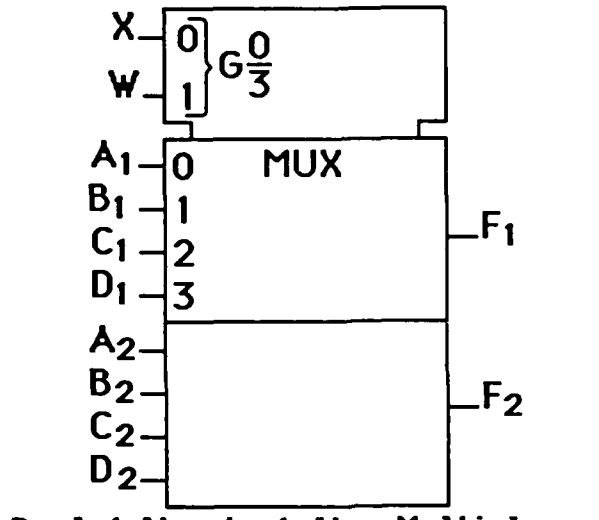
Table 1. Design Verification Vectors for Fig. 1 Multiplexer. (a) 8 vectors, (b) 20 vectors.

(a)

W	X	A_1	B_1	C_1	D_1	A_2	B_2	C_2	D_2	F_1	F_2
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	1	1
0	1	0	1	0	0	0	1	0	0	1	1
1	0	0	0	1	0	0	0	1	0	1	1
1	1	0	0	0	1	0	0	0	1	1	1

(b)

W	X	A_1	B_1	C_1	D_1	A_2	B_2	C_2	D_2	F_1	F_2
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	1	1
0	1	1	0	0	0	1	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	1	0	0	1	1
1	0	0	1	0	0	0	1	0	0	0	0
1	1	0	1	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0	0
0	1	0	0	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	0	1	0	1	1
1	1	0	0	1	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	0	0	1	0	0
1	0	0	0	0	1	0	0	0	1	0	0
1	1	0	0	0	1	0	0	0	1	1	1



Dual 4-line to 1-line Multiplexer
(153 or HEF 4539B)

Figure 1. Dual 4-line to 1-line multiplexer (54153 or 4539B without enable circuits).

Definition: By Boolean testing is meant the application of a sequence of input patterns to a physical digital device (typically a die on a silicon wafer or a packaged integrated circuit) and the observation of the output response patterns with the objective of determining whether or not the device is broken.

This definition refers to input and output patterns in order to rule out the use of specific voltages, currents or waveforms to determine device parameters. Such measurements correspond to parametric rather than Boolean testing. Parametric testing is discussed in [Camenzind 72] and [Chrones 81].

FUNCTIONAL TESTING

The term Boolean testing is used here to describe the test procedure that is called "functional testing" by the manufacturers of automatic chip test equipment. This usage is illustrated by the following quote from pp. III-A-3,4 of [Doyle]:

> For example, a TTL bipolar NOR gate may switch properly and show the proper output voltage levels when the inputs are toggled, but it may require significantly higher input current than normal to produce the change. In this example, the TTL gate is 'functional' in the sense that the truth table could be satisfied. On the other hand, the sample failed one set of 'parametric' tests. That is, at one specific temperature, power supply voltage, current, etc., the unit failed because of high input current. By the same token, this sample may well pass all tests at, for instance, a lower supply voltage or a different temperature.

Generally speaking, it is impossible to completely separate functional from parametric testing.

The reason for introducing a new term is the existence of alternate definitions of the functional testing terminology. Only integrated circuit chip testing is discussed here — board test is closely related and is also very important. The expression "functional testing" is used in connection with board test to specify that the test signals are applied through the board I/O edge connectors. The other major type of board test is called "in-circuit testing" or "bed-of-nails testing." This refers to the use of a special fixture that allows access to each node of the board and thus allows each component to be tested individually. Automatic test equipment for board test is discussed in [Stover 84].

There is another use (or misuse) of the expression "functional testing." It is sometimes used to describe a test that detects failures by "verifying correct operation" rather than by verifying the absence of specific faults. This usage is based on the desire to generate test patterns without reference to a specific implementation or to a model of which faults are detected by the tests. It is not uncommon to use the design verification vectors for such a "functional test."

The effectiveness of using the design verification vectors as test vectors can be illustrated with the Fig. 2 implementation of the Fig. 1 multiplexer. The 20-vector set of design verification vectors (Table 1b) detects all single-stuck faults, but is a rather uneconomical test since it is possible to detect all single-stuck faults with only the 8 vectors of Table 2.

The set of 8 design verification vectors (Table 1a) detects only 68% of the single-stuck node faults. Whether or not a specific fault coverage is acceptable can be approximated in terms of the process yield and the desired maximum defect level of the parts after testing. This determination is discussed in [Williams 85]. The defect level is shown to equal $1 - Y^{(1-T)}$ where Y is the process yield and T is the single-stuck fault coverage (fraction of single-stuck faults detected by the test set). For a 68% fault coverage and a process yield of 90%, the defect level is predicted to be approximately 33,000 DPM (defects per million). This is an unacceptably high defect level; current industrial practice aims at defects levels below 100 DPM.

Table 2. Single-stuck Fault Test Set for Fig. 2 circuit.

W	X	A_1	B_1	C_1	D_1	A_2	B_2	C_2	D_2	F_1	F_2
0	0	0	1	1	d	0	1	1	d	0	0
0	1	1	0	d	d	1	0	d	d	0	0
1	0	1	d	0	1	1	d	0	1	0	0
1	1	d	d	1	0	d	d	1	0	0	0
0	0	1	d	d	d	1	d	d	d	1	1
0	1	d	1	d	d	d	1	d	d	1	1
1	0	d	d	1	d	d	d	1	1	1	1
1	1	d	d	d	1	d	d	d	d	1	1

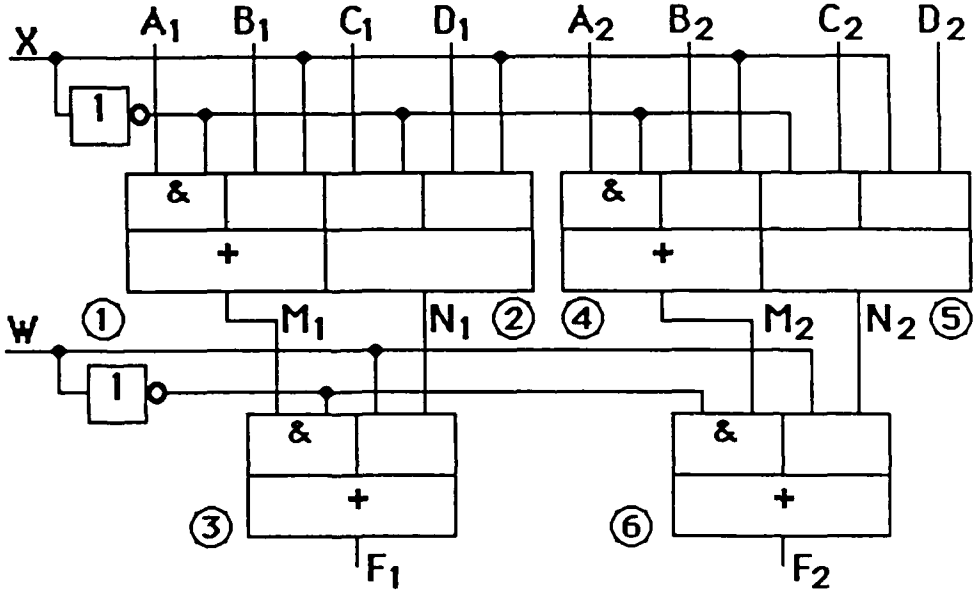


Figure 2. One implementation of the Fig. 1 multiplexer.

The fault coverage (68%) of the 8 design verification vectors was determined by a complete fault simulation. Since fault simulation can be very expensive (in computer time) for large circuits, simpler methods are often used to estimate the quality of a test set. One popular technique checks that each node of the circuit is "toggled," receives both a 0-signal and a 1-signal. The advantage of this approach is that only a fault-free rather than a fault simulation is necessary. Such Toggle Test Sets are discussed in [Hughes 85]. Even simpler are the Pin Test Sets that check only that the primary inputs and primary outputs are toggled.

The 8 design verification vectors toggle all nodes and thus satisfy the requirements of both the Toggle Test Set and the Pin Test Set. For the multiplexer example considered here, this shows that use of design verification vectors and either a Toggle Test Set or Pin Test Set validation could lead to the use of a test set with only 68% fault coverage. A more careful technique would be to do a fault simulation of the design verification vectors and than choose additional vectors until an acceptable fault coverage is obtained. By adding 4 additional vectors to the 8 verification vectors, it is possible to get 100% single-stuck fault coverage. Thus, the augmented verification vector set has 12 vectors instead of the 8 required for full single-stuck fault coverage.

This discussion of using design verification vectors for Boolean testing demonstrates some of the problems that can arise when test sets are generated without attention to the structure of the circuit being tested. In fact it was the lack of success with the generation of tests based on the functional operation of circuits that led to the development of the stuck-fault model and the use of structural information to generate test sets. The switch from "functional" tests to "structural" tests was signalled in a paper presented by R. Eldred at the August 1958 meeting of the ACM, [Eldred 59]. The opening sentence of this paper is:

> In order for the successful operation of a test routine to guarantee that a computing system has no faulty components, the test conditions imposed by the routine should be devised at the level of the components themselves, rather than at the level of programmed orders.

Because of the complexity and density of modern digital circuits, Boolean testing is now recognized as an important, perhaps limiting, factor in the continued progress in integrated circuit manufacture. Two aspects of Boolean testing are currently being reexamined:

> (1) The adequacy of the single-stuck fault model for VSLI and ULSI is being studied because other types of faults may become more important with the new circuits, and
>
> (2) techniques to generate tests without using the detailed structure of the circuit are being sought, either because detailed test pattern generation is becoming too expensive or because detailed structural information is not available for many catalog parts.

The following section discusses the necessity of a fault model for test pattern generation. Choices among various fault models are considered. A very restrictive fault model like single-stuck faults permits small test sets but can fail to detect many actual failures. Weaker fault models that represent more realistic failure modes are considered.

FAULT MODELS

It is, in general, impossible to determine the quality of a test set for a circuit without making some assumption about the faults that are to be detected by the test.

As an example to illustrate this statement, consider the 2-line to 1-line multiplexer of Fig. 3a. Since this is a 3-input combinational circuit, the exhaustive test set of Table 3 that contains all 8 valid input vectors should test the circuit completely. However, if the test set is applied in the order given in the table, it fails to detect the bridging fault shown in Fig. 3b. The bridging fault causes the topmost input to the multiplexer to be equal to Ay, the AND function of the A-input and the y-output. (Such a fault could not occur for all circuit realizations, but there are circuit configurations for which it is an accurate representation.)

Table 3. Exhaustive Test Set for Fig. 3 Circuit.

X	A	B	y
0	0	0	0
0	0	1	1
0	1	1	1
0	1	0	0
1	1	0	1
1	1	1	1
1	0	1	0
1	0	0	0

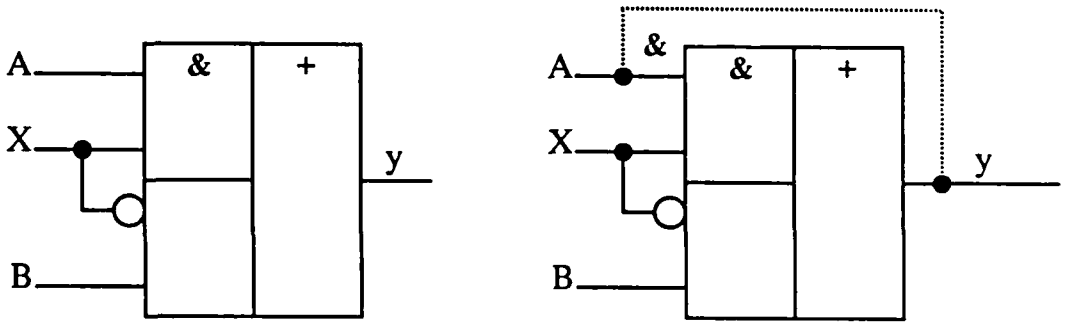


Figure 3. Two-line to one-line multiplexer. (a) Fault free, (b) With bridging fault.

The largest fault class for which it is practical to generate a test set is the class of *combinational faults*, faults that change the circuit function but do not increase the number of states in the circuit. If no limit is imposed on the number of extra states in the faulty circuit, then it is impossible to construct a complete test set, a test set that detects all faults that change the circuit function, [Moore 55]. Test set generation for "bounded sequential faults," faults that can increase the number of circuit states by no more than a fixed number of additional states, is possible in principle but not in practice. The qualifier — "that do change the circuit function" — is used above in order to allow for the presence of untestable faults in the circuit under test.

UNTESTABLE FAULTS

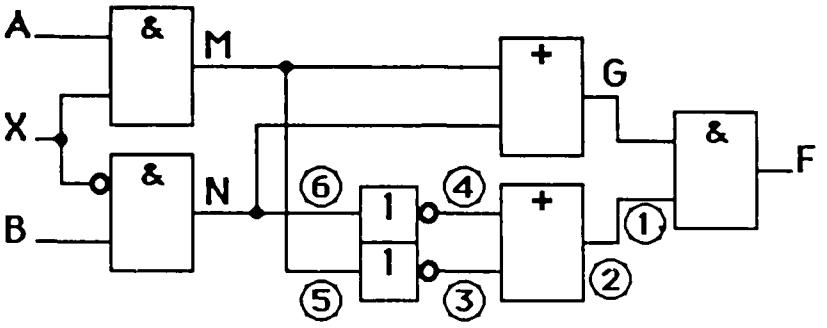
An untestable fault is a fault that does not affect the circuit function. Untestable faults are sometimes called undetectable faults. There are four mechanisms that can cause a fault to be untestable:

(1) Redundant circuitry — the fault can occur in some portion of the circuit that is redundant in the sense that it has no effect on the circuit function. The usual reason for the presence of redundancy is a design mistake. This situation is demonstrated in Fig. 4a. Stuck-at-1 faults on leads 1,2,3, or 4 are untestable as are stuck-at-0 faults on leads 5 or 6. Since either M or N is always 0, the output of the bottom OR gate (lead 2) is always 1. Neither the bottom OR gate, the two Inverters, nor the output AND gate are required and, in fact, the output function could be taken from lead G rather than F.

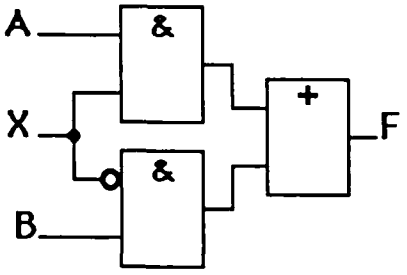
(2) Internal signal dependencies — there may be one or more internal gates at whose input leads some pattern or patterns cannot occur. For example, in Fig. 4b it is not possible to have a 1-signal on both inputs to the OR gate. In such a situation, any fault that changes the gate function for only the input combinations that never occur will, of course, have no effect on the entire circuit operation. In Fig. 4b an example of such a fault is one which changes the OR gate to an XOR gate. The presence of such faults cannot be detected by a Boolean test.

(3) Hazard control circuitry — if part of the circuit is present to eliminate a hazard, faults that disable the hazard control will not be detectable since they have no effect on the static circuit operation. In Fig. 4c the output H of the bottom AND gate is only 1 when both A and B are 1, in which case one of the other AND gate outputs must be 1 since X or X' must be 1. The bottom AND gate is present only to prevent a static logic hazard and cannot be detected by a Boolean test. For a discussion of hazards, see Chapter 3 in [McCluskey 86].

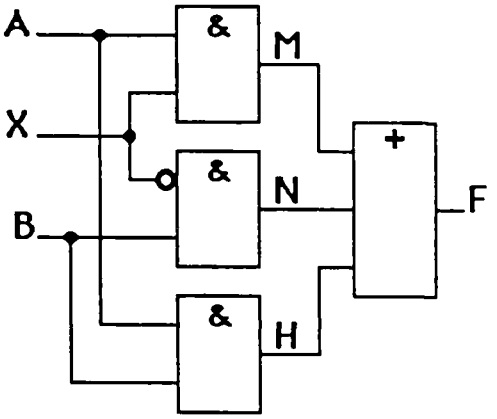
(4) Error control circuitry — Circuitry is sometimes added to a design in order to detect failures while the system is in use. The circuit in Fig. 5 is a 2-line to 4-line decoder with added error detection circuitry. Since exactly one of the decoder outputs is equal to 1, one of the OR gate outputs is always 1 and the XOR gate output (E) is always 1. Any decoder failure that causes all outputs to be 0 will change E to 0 since both inputs to the XOR gate will be 0. Similarly any failure that makes both inputs to the XOR gate equal to 1 will change E to 0. When the decoder circuitry is fault-free, the E output is always 1, so that the fault E stuck-at-1 is undetectable.



(a)



(b)



(c)

Figure 4. Three implementations of a 2-line to 1-line multiplexer illustrating untestable faults. (a) Redundant circuitry, (b) Signal dependency, (c) Hazard control circuitry.

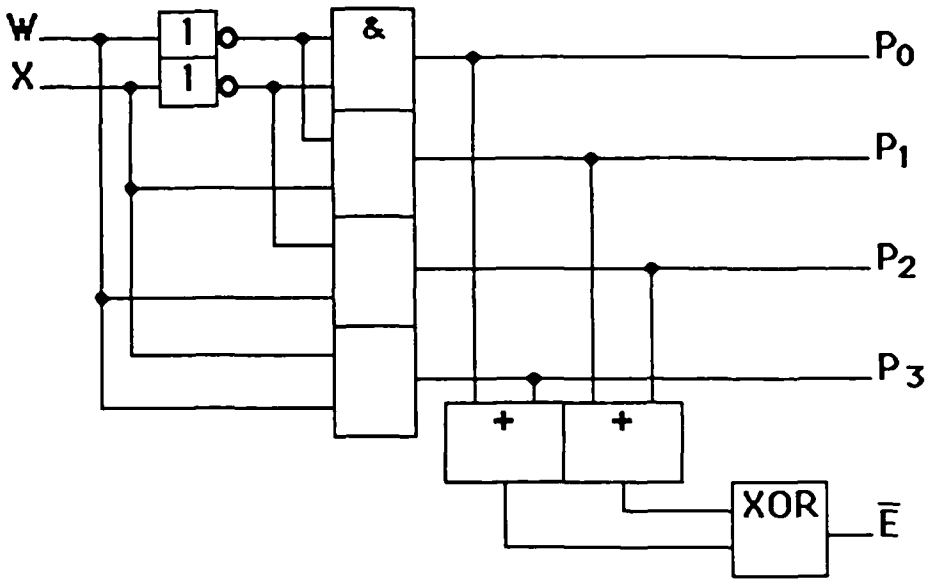


Figure 5. 2-line to 4-line decoder with error detection circuits.

EXHAUSTIVE TEST SETS

In order to test a circuit for the presence of combinational faults, an exhaustive test set is required.

Definition: An Exhaustive Test Set for an n-input combinational circuit is the set of all 2^n possible input vectors. For a sequential circuit, an exhaustive test set is a checking experiment, [Friedman 71].

The exhaustive test set for the Fig. 1 multiplexer consists of all 10-bit patterns, 1024 of them, since there are 10 inputs to this circuit. Only combinational circuits are considered here. Sequential circuit testing is important, but it appears uneconomic to test large sequential circuits without using a scan path technique to permit the use of combinational circuit tests, [McCluskey 86].

Theorem. The smallest test set that detects any combinational fault in an n-input combinational circuit is the 2^n pattern exhaustive test set.

Proof. Suppose a shorter test set is used. Then at least one input pattern is not applied. The combinational fault that changes the circuit function for only this input pattern will not be detected. Suppose all input patterns are applied, then any combinational fault must change the circuit output for at least one pattern causing the fault to be detected.

There are two difficulties with exhaustive test sets:

(1) There are two classes of physical failures that can cause non-combinational faults. The bridging fault of Fig. 3b is an example of one such fault class that can fail to be detected by an exhaustive test. Failures in which two points in a circuit are incorrectly connected together are not uncommon. Such failures can be modelled as bridging faults, [Mei 74]. Some bridging faults can introduce additional states into a circuit.

The other important non-combinational faults, stuck-open faults, occur in MOS circuits. They are caused by charge storage on nodes that have some of their connections removed due to open-circuit failures, [Wadsack 78]. A lot of attention is now directed towards detecting MOS stuck-open faults. One approach derives tests for specific MOS stuck-open faults and adds these to a stuck-fault test set, [Reddy 83], [Jha 85]. This process is expensive in the amount of required computation. Another method adds circuitry to the MOS circuit to facilitate testing for MOS stuck-open faults, [McCluskey 81], [Liu 87]. The cost penalty for this technique is the area overhead of the additional circuits. The most promising method when the layout can be controlled is to design the gate geometry so as to reduce the possibility of a MOS stuck-open fault to an acceptably low value, [Zasio 85], [Koeppe 87]. A recent study to determine whether stuck-open faults occurred in significant numbers in production CMOS chips did not detect any stuck-open faults, [Turner 85]. This suggests that stuck-open faults may not be a concern for well-designed CMOS chips.

Bridging faults have received much less attention than MOS stuck-open faults, but there seems to be no justification for this. The bridging faults appear to be, if anything, more common than the MOS stuck-open faults. It is possible to generate tests for specific bridging faults and add these to a stuck-fault test set. Another approach would be to control the layout to reduce the probability of non-combinational bridging faults.

(2) The other difficulty with exhaustive test sets is that they can be very long. For an n-input combinational circuit, all 2^n input patterns are required for an exhaustive test.

If the number of inputs to a circuit is so large that exhaustive testing is not feasible, it is necessary to adopt a more restrictive fault model than the combinational fault model.

PSEUDO-EXHAUSTIVE TEST SETS

For circuits in which each circuit output depends on a subset of all the inputs, it may be feasible to test each output exhaustively even though the entire circuit is not tested exhaustively. This technique is based on a fault model, the constrained dependence fault model, which assumes that the set of inputs on which a circuit output depends is not increased by the presence of the fault. The corresponding test set that applies all input combinations of each set of inputs on which a circuit output is dependent is called a pseudo-exhaustive output function verification test set, [McCluskey 82A,B, 83, 84A,B]. Since each of the outputs in the Fig. 2 multiplexer depends on 6

inputs, each output can be tested exhaustively with 64 patterns. It is possible to test both outputs at the same time, using only 64 patterns, by using the pseudo-exhaustive test method.

For some circuits, this output function verification test may be too lengthy. A further restriction on the fault class test is then necessary. The circuit must be then segmented into subcircuits, each of which is tested exhaustively. This process is called pseudo-exhaustive segment function verification and is described in [McCluskey 81]. For the Fig. 2 multiplexer circuit this type of test can be done with 16 (rather than 64) patterns. Each of the six AND-OR circuits labelled in the figure is tested exhaustively.

VERIFICATION TESTING

Most combinational networks have more than one output. In many cases each of the outputs depends on only a subset of the inputs. For example, the parity generator network of the TI SN54/74LS630 (shown in Fig. 6) has 23 inputs and 6 output functions, but each output depends on only 10 of the inputs. It may not be practical to exhaustively test the outputs by applying all combinations of the network inputs (2^{23} for the example). However, it may be possible to exhaustively test each output by applying all combinations of only those inputs on which the output depends. For the SN74LS630, each output can be exhaustively tested with 2^{10} =1024 input patterns and all six outputs tested one after another with (6)(1024)=6144 patterns. In fact, for this circuit it is possible, by an appropriate choice of input patterns, to apply all possible input combinations to each output concurrently rather than serially. Thus, with only 1024 rather than (6)(1024) test patterns, each output can be tested exhaustively by using the verification testing techniques described in [McCluskey 84B] and [Barzilai 81].

The SN74LS630 is an example of a circuit for which it is possible to test all outputs concurrently by applying (to the entire circuit) only as many patterns as are necessary to exhaustively test one of the outputs. Two inputs that never appear in the same output function can have the same test signal applied to both. (Applying the complement test signal to one of the two inputs is preferable since this does not increase the cost of the test and increases the bridging fault coverage.) This fact can be used to reduce the number of required test signals. If the number of required test signals is equal to the maximum number of inputs upon which any output depends, the circuit is called a *maximal test concurrency circuit*, MTC circuit.

An example of a very simple non-MTC circuit with three inputs and two outputs is shown in Fig. 7. The f output depends only on inputs w and x, while the g output depends on x and y. It is possible to apply the same test signal to both w and y since no output depends on both of these inputs. The four input patterns shown in the figure are such that all possible combinations of values are applied to w and x, and also all combinations of x and y are present. Thus, both outputs f and g are tested exhaustively by only four input patterns rather than the eight patterns required for full exhaustive test.

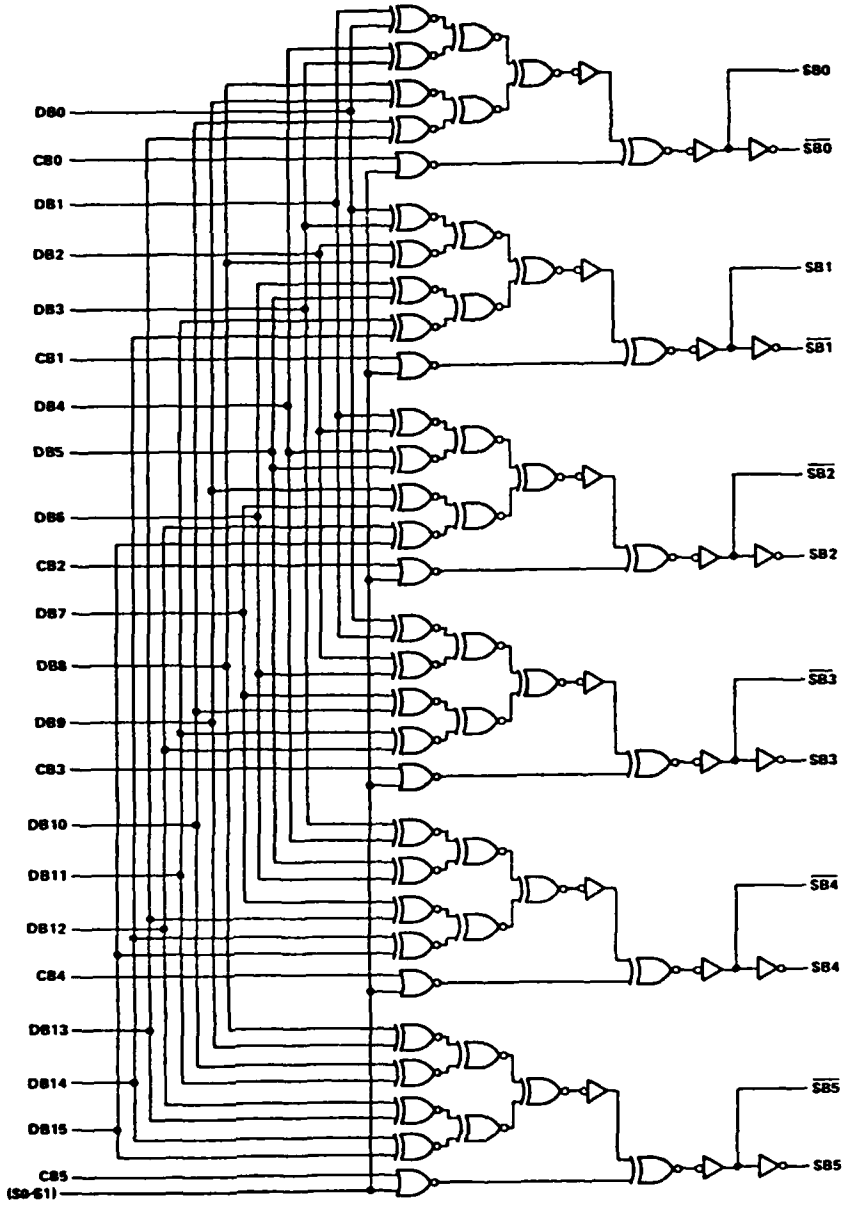


Figure 6. Parity generator network of the TI SN54/74LS630.

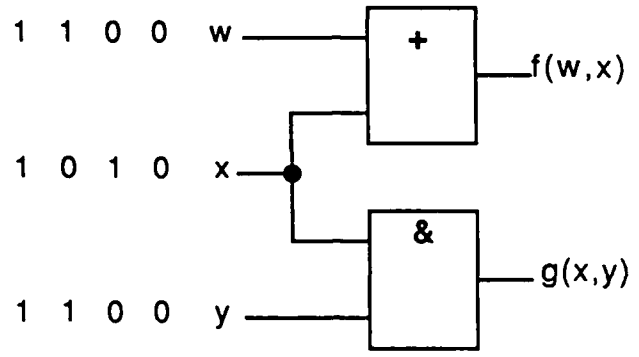


Figure 7. Simple example of an MTC circuit with verification test inputs.

Figure 8 shows a very simple example on a non-MTC circuit. There are three inputs and each possible pair occurs in some output function. Thus, no two inputs can have the same test signal applied to both. Three test signals are required; however, it is possible to test each of the output functions exhaustively with only four test patterns. All outputs are tested concurrently with the same number of test patterns that are necessary to test one of the outputs. This is an example of the general situation of an n-input circuit in which each output depends on at most n-1 inputs. Such circuits can always have all outputs tested concurrently with at most 2^{n-1} test patterns. The appropriate patterns are all n-bit vectors having the same parity (either all even parity as in Fig. 8 or all odd parity).

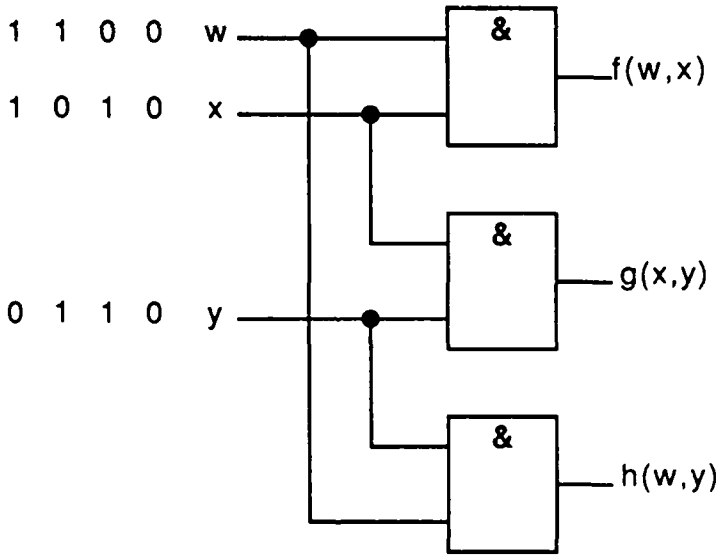


Figure 8. Simple example of a 3-input, 3-output non-MTC circuit with verification test inputs.

A simple example of a non-MTC circuit for which it is not possible to verify (test concurrently) all outputs with the same number of patterns required for testing a single output is shown in Fig. 9. In this circuit, each of the 6 outputs depends on 2 of the 4 inputs. All 6 outputs are verified with the 5 input patterns shown on the figure. To test the circuit exhaustively would require 16 patterns. Testing exhaustively each of the output functions in succession takes (4)(6)=24 patterns. It is possible to test pairs of outputs such as f1 and f6 at the same time since they depend on disjoint sets of inputs. This strategy requires 3 tests of 4 patterns each or 12 total patterns. The numbers of inputs and outputs used in this example are too small to be significant. However, it does serve to illustrate the percentage reduction in test length obtainable with verification test techniques.

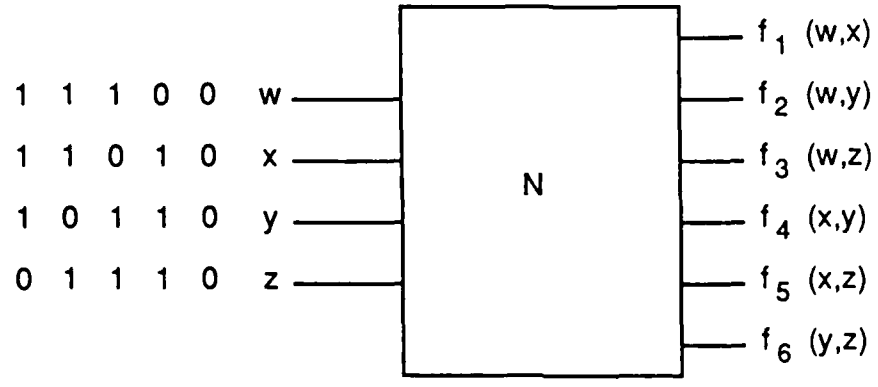


Figure 9. Simple example of a 4-input, 6-output non-MTC circuit with verification test inputs.

It has been shown in [McCluskey 84B] that any network for which no output depends on all inputs can be tested pseudo-exhaustively with fewer than 2^n test patterns. This paper also derives a specification for test sets that consist of constant-weight vectors. Theoretical results on the use of constant-weight vectors are derived in [Tang 83] and on the minimum number of vectors for an output verification test set in [Chandra 83]. Since constant-weight counters are not, in general, easy to realize economically, several schemes to use linear feedback shift registers have been developed. These are discussed in the section on implementations.

If any output depends on too many inputs (a typical bound is 20 since this corresponds to a test length of approximately one million vectors), output verification is not sufficient as a pseudo-exhaustive test. For such a network, the network (or that portion that cannot be tested using output verification) must be tested using the techniques described in the section on segmentation testing.

For output verification testing, the objective is to calculate a verification test set, which is defined as follows.

Definition: A *Verification Test Set* (VTS) for a network N is a set of network input patterns. These patterns have the property that the subset I_j of the network inputs, upon which output f_j depends, has all possible combinations of values applied. This must be true for all j.

DEPENDENCE MATRIX

The only information about the network N required to determine the VTS is the input sets I_j. The specific functional dependence of ouputs on inputs is not required. The required information is conveniently represented by means of the dependence matrix D(N).

Definition: The *dependence matrix* (D(N)) for a network N has m rows and n columns. Each row represents one of the network outputs and each column one of the network inputs. An entry is 1 if and only if the corresponding output depends on the corresponding input. All other entries are 0. The dependence matrices for Figs. 6 and 7 are shown in Table 4.

Table 4. Dependence Matrices for Figs. 6 and 7. (a) D(N) for 74LS630 Network of Fig. 6, (b) D(N) for Fig. 7.

(a)

	DATA BITS																CHECK BITS							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	S0.S1
SBO	1	1	0	1	1	0	0	0	1	1	1	0	0	1	0	0	1	0	0	0	0	0	1
SB1	1	0	1	1	0	1	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	1
SB2	0	1	1	0	1	1	0	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1
SB3	1	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	1	0	0	1
SB4	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1
SB5	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1

(b)

	w	x	y
f	1	1	0
g	0	1	1

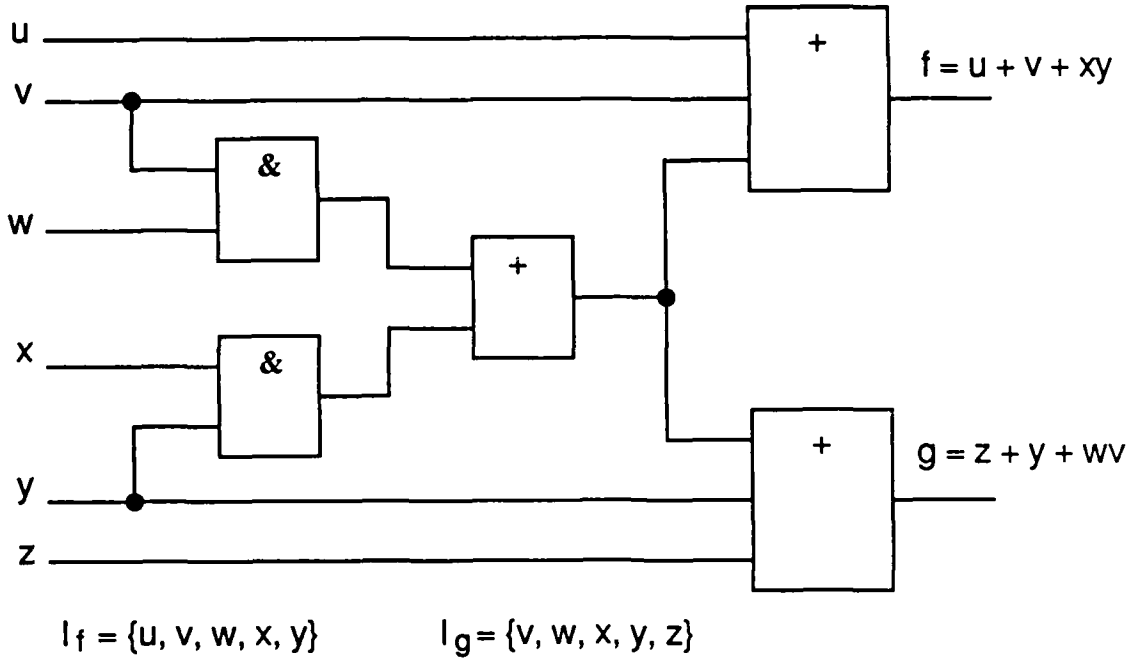


Figure 10. Network to illustrate calculation of I_j.

The dependence matrix is simply calculated by *tracing paths* from outputs to inputs. There is the possibility that this method can result in a situation in which the matrix indicates that an output depends on some inputs for which there is no functional dependence due to the logic realized by the network. For example, in the network of Fig. 10, path tracing would indicate that f depends on u,v,w,x,y, while there is, in fact, no functional dependence of f on w.

Better fault coverage is obtained by using structural rather than functional dependence. An output is more likely to be affected by a fault in a variable on which the output is structurally dependent than a fault on which the output has no dependence at all. If the structural information is available there can be two advantages in using the structural rather than the actual functional dependency: better fault coverage and less computation. In some situations, only the functional dependence information is available. The dependence matrix can be constructed directly from this information without any consideration of the structure.

Lemma 1. Let x_i and x_j be two inputs of N such that there is at least one output that depends on both x_i and x_j. Then in any verification test set for N, x_i and x_j must be assigned all 4 possible pairs of values.

Lemma 2. Let $A_s = [x_i, x_j, \ldots x_k]$ be a subset of the inputs with the property that no two of these inputs are both present in any I_j. Thus no output depends on more than one member of A_s. A VTS exists in which the members of A_s always have the same value assigned to all of them for each input pattern.

The first step in finding a VTS is to determine the subsets As which can have common inputs. This is done by partitioning the columns of the dependence matrix D(N) to form a partitioned dependence matrix.

Definition: The *partitioned dependence matrix* (DP(N)) corresponding to D(N) is formed by partitioning the columns of D(N) into sets such that:

(i) Each row of a set has at most one 1-entry, and

(ii) the number of sets, p, is a minimum.

There is a corresponding partition, P, of the network inputs in which all inputs belonging to the same set of columns of DP(N) are included in the same class A_s. To determine the VTS for a network, N, it is first necessary to obtain a partition, P. For each pattern in the VTS, all inputs of the same class of P are assigned the same value. Table 5a illustrates this. The DP matrices corresponding to Table 4 are shown in Table 5.

Table 5. Partitioned Dependence Matrices for Figs. 6 and 7. (a) DP(N) for Fig. 6, (b) DP(N) for Fig. 7, (c) VTS for Fig. 7, (d) RVTS for Fig. 7.

(a)

	DATA BITS												CHECK BITS							
	0	15	1	14	3	12	4	11	8	7	9	6	10	5	13	2	0	1	2	3	4	5	S0	S1
SB0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	1	
SB1	1	0	0	1	1	0	0	1	1	0	0	1	0	1	0	1	0	1	0	0	0	0	1	
SB2	0	1	1	0	0	1	1	0	0	1	1	0	0	1	0	1	0	0	1	0	0	0	1	
SB3	1	0	1	0	0	1	0	1	0	1	0	1	1	0	0	1	0	0	0	1	0	0	1	
SB4	0	1	0	1	1	0	1	0	0	1	0	1	0	1	1	0	0	0	0	0	1	0	1	
SB5	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0	0	0	0	0	0	1	1	

```
P = (DB0,DB15)(DB0,DB15)(DB0,DB15)(DB0,DB15)(DB0,DB15)(DB0,DB15)
    (DB0,DB15)(DB0,DB15)(CB0,CB1,CB2,CB3,CB4,CB5)(S0.S1)

          p = 10, W = 10
```

(b)

```
        w  y  x
f      |1  0 |1|
g      |0  1 |1|

P = (w,y)(x), p=2, W=2
```

(c)

```
w y x
0 0 0
0 0 1
1 1 0
1 1 1
```

(d)

```
  w
y x
0 0
0 1
1 0
1 1
```

There may be more than one DP(N) corresponding to a D(N) matrix, as is illustrated in Table 6.

Table 6. Example of D(N) for which PD(N) is not Unique. (a) D(N), (b) DP(N).

(a)

1	2	3	4	5	6
1	0	0	1	0	1
0	1	0	1	0	0
0	0	1	0	1	0

(b)

1	2	3	4	5	6
1	0	0	1	0	1
0	1	0	1	0	0
0	0	1	0	1	0

P = (1,2,3)(4,5)(6)
n=6, p=3, W=3

or

1	2	3	4	5	6
1	0	0	1	0	1
0	1	0	1	0	0
0	0	1	0	1	0

P = (1,2)(3,4)(5,6),
6>3=3

Definition: Let *W* be the *maximum row weight* (maximum number of 1s in any row of N).

Lemma 3. For any N it is true that $n \geq p \geq W$.

Table 6 shows an example for which $n > p = W$, and a matrix for which $n > p > W$ is shown in Table 7.

Table 7. A DP(N) Matrix for which $n > p > W$.

1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1

n = 4, p = 3, W = 2

VERIFICATION TEST SET GENERATION

It follows from Lemmas 1 and 2 that a Verification Test Set can be derived in which all inputs corresponding to the same partition of DP(N) have the same test signal applied and any two inputs from different partitions have distinct test signals applied. In the following discussion, only verification test sets in which all inputs of a partition receive the same test signal will be considered. It is thus convenient to use a reduced partitioned dependency matrix. In actual practice, it is often advantageous to apply the complement of a test signal rather than the test signal itself to some of the inputs from the same partition. The advantages of this arrangement are increased coverage of bridging faults between two different partitions and more equal loading on the stages of the test pattern generation circuitry.

Definition: The *reduced partitioned dependency matrix* (RDP(N)) is obtained from DP(N) by replacing each column of DP(N) by a single column in which a row has a 1 iff any column of the corresponding row of DP(N) has a 1. DP(N) has m rows and p columns.

To avoid having to write identical columns many times, a reduced verification test set will be specified.

Definition: A *reduced verification test set* (RVTS) is obtained from a verification test set by removing all repetitions of identical columns. Each column of an RVTS will typically correspond to one class of network inputs.

An example of an RVTS is shown in Table 5d.

Theorem 2. If p = W, the minimum-row RVTS has 2^W rows and each possible row appears exactly once. The corresponding circuit is a maximum test concurrency circuit, MTC.

This is illustrated in Table 5c. Also, Table 5a shows that p=W=10 for the 74LS630 network of Fig. 7. The 74LS630 parity generator can thus be tested with a minimum-length verification test set having 2^{10} patterns. This is a considerable reduction from the 2^{23} patterns required for classical exhaustive testing. Each of the six outputs could also be tested exhaustively in sequence with a total of (6)(1024) patterns. This strategy requires a longer, more complex test application procedure and response analysis technique than the minimum-length verification test set.

Each row of the RPD(N) matrix corresponds to one circuit output. The columns in which the row has 1s correspond to the input test signals upon which that output depends. The following conditions on the RVTS follow directly from these observations.

Theorem 3. Suppose that V is a t by p matrix representing a RVTS for RDP(N). Corresponding to each row r_i of RDP(N) define a submatrix V_i of V which consists of only those columns of V which correspond to columns of RDP(N) in which ri has a 1. Then if V represents a RVTS for RDP(N) each submatrix Vi has each possible binary combination appearing at least once.

This is illustrated in Table 8. The submatrix V_2 that consists of columns a,b, and d of Table 8b and corresponds to row 2 of Table 8a is shown in Table 8c. All eight possible combinations of three bits occur in Table 8c, thus satisfying the conditions of Theorem 3. Note that the row 11001 is missing from Table 8a and that the corresponding columns — a,b, and e — of Table 8b do not contain all combinations (000, 101, 011, and 110 are missing).

Table 9 shows a V matrix for which any 3 columns contain all binary combinations. This matrix is a Universal RVTS, a RVTS for any RDP(N) having p=5 and W=3. It is thus possible to use the V matrix of Table 9 as a RVTS for Table 8a in place of the V matrix of Table 8b. The disadvantage of using Table 9 is that it contains two more rows than Table 8b. The advantage is that V matrices such as Table 9 are much easier to construct and generate electronically than those such as Table 8b. Table 8 was included to illustrate the general possibilities for RVTS construction. Only

matrices such as that of Table 9 will be considered in the following discussion. The ease of construction and generation of such matrices is felt to outweigh the disadvantage of their lack of minimality for specific networks. For self-test designs, the general matrices such as Table 9 are of greatest importance.

Table 8. An example of a Verification Test Set. (a) RDP(N), (b) V, (c) V_2.

(a)

	a	b	c	d	e
1	1	1	1	0	0
2	1	1	0	1	0
3	1	0	1	0	1
4	1	0	0	1	1
5	0	1	1	1	0
6	0	1	1	0	1
7	0	1	0	1	1
8	0	0	1	1	1

(b)

a	b	c	d	e
1	0	0	0	0
0	0	0	1	1
0	0	1	0	1
1	0	1	1	0
1	1	0	0	1
0	1	0	1	0
0	1	1	0	0
1	1	1	1	1

(c)

a	b	d
1	0	0
0	0	1
0	0	0
1	0	1
1	1	0
0	1	1
0	1	0
1	1	1

Table 9. A Universal Verification Test Set for p=5 and W=3.

	a	b	c	d	e
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1
6	0	1	1	1	1
7	1	0	1	1	1
8	1	1	0	1	1
9	1	1	1	0	1
10	1	1	1	1	0

UNIVERSAL REDUCED VERIFICATION TEST SETS

Definition: A p-column matrix is a universal reduced verification test set matrix, U(p,w), iff all submatrices of w columns (and all rows) of U(p,w) contain all possible combinations of w binary digits.

The matrix of Table 9 is a U(5,3) matrix.

Lemma 3. Any U(p,w) matrix is also a U(p,j) matrix for all j less than w.

Lemma 4. A U(p,w) matrix specifies a RVTS for any network with the same value of p and W=w.

Theorem 4. For p = W + 1, the minimum-row U(p,W) = U(W+1,W) has 2^W rows. Two types of U(W+1,W) exist: one has all possible even-parity rows and the other all possible odd-parity rows. This is illustrated in Table 10. Figure 8 also shows a constant-parity RVTS.

Table 10. Example of RVTS when p = W + 1. (a) DP(N), (b) Even-parity RVTS, (c) Odd-parity RVTS.

(a)

1	2	34
1	1	00
1	0	10
1	0	01
0	1	10
0	1	01

(b)

1	2	3 4
0	0	0
0	1	1
1	0	1
1	1	0

(c)

1	2	3 4
0	0	1
0	1	0
1	0	0
1	1	1

Proof. A constant-parity U(w+1,w) has w+1 columns and 2^W different rows. If one column is removed, there are 2^W different w-bit rows. (These produce an exhaustive test on the corresponding p inputs.) The bits of the removed column are determined by the requirement that each w+1 bit row have fixed parity. Thus, if two w-bit rows were identical, the corresponding w+1 bit rows would be identical. This isn't possible because of the definition of the U(w+1,w). The constant-parity U(w+1,w) has a minimum number of rows since 2^W rows are required to have all patterns occur in w columns.

When p > W+1, more than 2^p patterns are required for verification testing of the network. In fact, finding the minimum set of verification test patterns when p > W+1 is a difficult combinatorial problem, which appears to have been solved exactly only for the case of W = 2, [Chandra 83]. Fortunately, finding the absolute minimum test set is of more theoretical than practical interest. Systematically derived test sets are most useful for actual testing applications. It is possible to derive universal verification test sets as sets of constant-weight vectors. Table 11 lists universal verification test sets for values of k from 2 to 10, [McCluskey 82b]. Each U(p,w) is made up of submatrices containing all possible rows of p columns having a specific weight (number of 1s). For example, the U(5,3) of Table 11 contains all possible rows of weight 1 or 4. The notation used in Table 11 for specifying U(p,w) is p[i,j,k], which represents a p column matrix containing all possible rows of weight i, j, or k. The notation B(i,j) specifies the binomial coefficient of i things taken j at a time. Only values of p>w+1 are listed in Table 11. When p=w, the U(p,w) is simply all

binary combinations of p bits. For p=w+1, the U(p,w) is all constant parity p-bit rows. Proofs of the entries in Table 11 can be found in [Tang 83].

The rows of Table 11 for even values of w have two entries for U(p,w), both of which have the same length. The second entry can be obtained by complementing (interchanging 0s and 1s) the first entry. Odd values of w produce minimum-row matrices that are self-complements. Since all combinations must be present in the submatrices, 0 and 1 can be interchanged without disturbing the properties of the matrix.

Lemma 5. If p[i,j,k] represents a U(p,w) matrix, then so does p[p-i,p-j,p-k].

Table 11. Specifications for U(p,w).

w	U(p,w)		Range	Number of Tests (rows)
2	p[0,p-1]	p[1,p]	p > 3	p+1
3	p[1,p-1]		p > 4	2p
4	p[1,p-2]	p[2,p-1]	p > 5	1/2 p(p+1)
5	p[2,p-2]		p > 6	p(p-1)
6	p[2,p-3]	p[3,p-2]	p > 8	B(p+1,3)
6	p[1,4,7]		p = 8	1/2 p(p+1)
7	p[3,p-3]		p > 9	2 B(p,3)
7	p[0,3,6,9]		p = 9	170
8	p[3 , p-4]	p[4 , p-3]	p > 11	B(p+1,4)
8	p[0,3,6,9]	p[1,4,7,10]	p = 10	341
8	p[0,4,8]	p[3,7,9]	p = 11	496
9	p[4,p-4]		p > 12	2 B(p,4)
9	p[1,4,7,10]		p = 11	682
9	p[0,4,8,12]		p = 12	992
10	p[4,p-5]	p[5,p-4]	p > 14	B(p+1,5)
10	p[1,4,7,10]	p[2,5,8,11]	p = 12	1365
10	p[0,4,8,12]	p[1,5,9,13]	p = 13	2016
10	p[0,5,10]	p[4,9,14]	p = 14	3004

Consider a circuit with n inputs and B(n,w) outputs in which each output depends on a different subset of w of the circuit inputs. Such a circuit represents a "worst case" circuit with limited dependency of outputs on inputs. A comparison of the test lengths required for the 54LS630 using different techniques is given in Table 12.

Table 12. Comparison of Different Test Lengths for Worst-Case Circuit.

Technique	Test Length		
	n = p, w	w = 10	n=23, w=10
Exhaustive Test	2^n	2^n	8,388,608.
Universal Verification Test	U(p,w)	B(n+1,5)	42,504.
Minimum Verification Test	———	———	1,024.

For BIST applications, the test sets specified in Table 11 could be implemented with constant-weight counters. It is also possible to use LFSRs (linear feedback shift registers) to generate verification test sets. The LFSR test sets are longer than the constant-weight test sets, but require fewer components. These issues are discussed further in the section on implementations.

SEGMENT VERIFICATION TESTING

It is possible that an output verification test set for a particular circuit is too long. This could be because some outputs depend on all inputs and 2^n is too large; or because the output verification test set, even though smaller than 2^n, is too long. In other words, there are circuits for which the output verification test approach does not result in a satisfactory test procedure. A pseudo-exhaustive test is still possible for such circuits, but it is necessary to resort to a partitioning or segmentation technique, [Bozorgui-Nesbat 80], [McCluskey 80,81]. The circuit is segmented into two or more subcircuits, and each segment is tested exhaustively.

Figure 11 shows Circuit A, a simple 6-input, 2-output circuit which will be used to illustrate this technique. An exhaustive test of this circuit takes 64 test vectors since n = 6. Output verification can be done with 32 vectors since F_1 depends on 5 inputs. Two segments are shown on the figure: Segment G with inputs W,X,Y and output g; and Segment H with inputs U,V,g,Z and ouputs F_1 and F_2.

In order to test segment G exhaustively, all 8 combinations must be applied to inputs W,X,Y and the response at g must be observed. Application of the input combinations can be done directly since all of the segment inputs are circuit primary inputs. Observation of the response requires that g be observable at a circuit primary output. This can be accomplished by using a multiplexer to connect g to a circuit output during test or by setting the other primary inputs to the circuit so as to sensitize a path from g to an output. By setting Z = 1, a path is sensitized from g to F_2. This arrangement is shown in Figure 12, along with the corresponding test patterns.

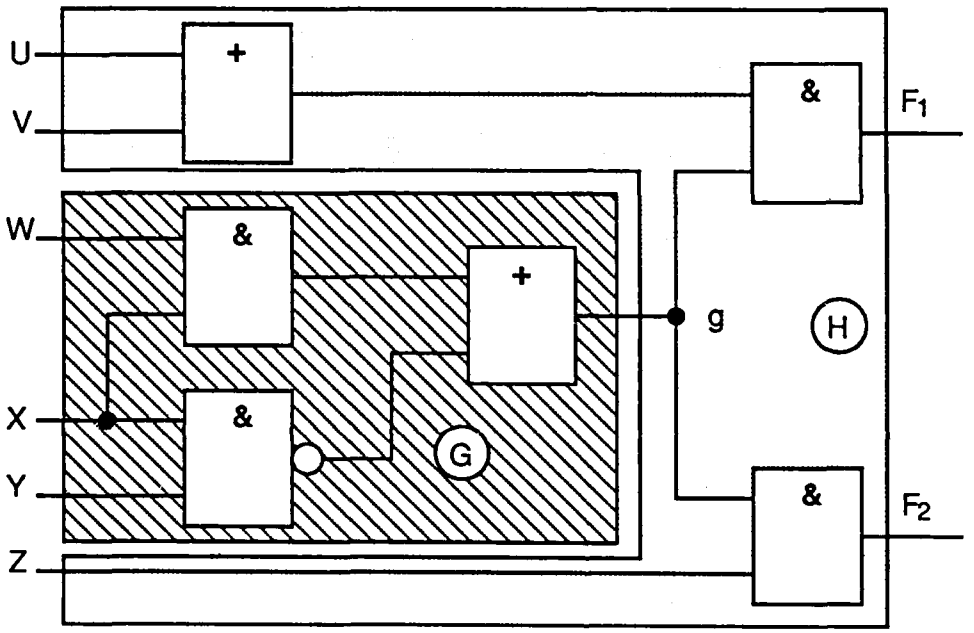
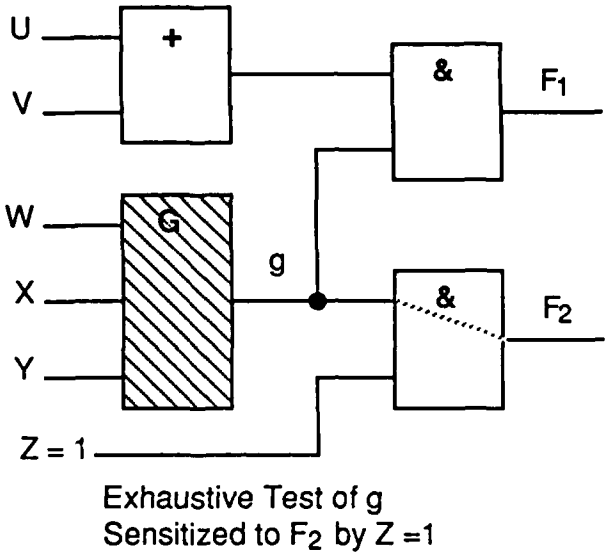


Figure 11. Circuit A, a simple example circuit with two segments shown.



	U V	W X Y	Z	g	F_1	F_2
1		0 0 0	1	1		1
2		0 0 1	1	1		1
3		0 1 0	1	1		1
4		0 1 1	1	0		0
5		1 0 0	1	1		1
6		1 0 1	1	1		1
7		1 1 0	1	1		1
8		1 1 1	1	1		1

Figure 12. Exhaustive test of segment G of Circuit A.

The exhaustive test of segment H requires that its inputs, U,V,g, Z be cycled through all 16 combinations. These are shown in Table 13a along with the patterns for segment G. The 24 patterns of Table 13a can be compressed since patterns 4 and 13 can be combined by filling in the unspecified entries in pattern 4 to make it identical with the specified entries of pattern 13. Similar combinations are possible for pattern pairs (5,21), (6,22), (7,23), (8,24). This compression results in the 19 patterns shown in Table 13b. Since segment H is a multi-output circuit, it is possible to test it with an output verification test rather than a full exhaustive test. This technique requires the 12 patterns shown in Table 13c.

Table 13. Pseudo-Exhaustive Test Sets for Circuit A. (a) Exhaustive Test Sets for Segments G and H, (b) Compression of Test Sets of a, (c) Exhaustive Test Set of Segment G and Output Verification Test Set of Segment H.

(a)

	G								H																
										1	1	1	1	1	1	1	1	1	2	2	2	2	2
	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4
U	-	-	-	-	-	-	-	-	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
V	-	-	-	-	-	-	-	-	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
W	0	0	0	0	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
X	0	0	1	1	0	0	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Y	0	1	0	1	0	1	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Z	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
g	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
F_1	-	-	-	0	-	-	-	-	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1
F_2	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

(b)

	G								H										
										1	1	1	1	1	1	1	1	2	
	1	2	3	4	5	6	7	8	9	0	1	2	4	5	6	7	8	9	0
U	-	-	-	0	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1
V	-	-	-	0	0	0	1	1	0	0	1	1	0	1	1	0	0	1	1
W	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
X	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0
Y	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0
Z	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	0	0	0	0
g	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1
F_1	-	-	-	0	0	1	1	1	0	0	0	0	0	0	0	0	1	1	1
F_2	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

(c)

	G								H		
										1	1	1
	1	2	3	4	5	6	7	8	9	0	1	2
U	-	-	-	1	0	1	0	1	0	1	0	-
V	-	-	-	1	0	0	1	1	0	0	1	-
W	0	0	0	0	1	1	1	1	0	0	0	1
X	0	0	1	1	0	0	1	1	1	1	1	1
Y	0	1	0	1	0	1	0	1	1	1	1	1
Z	1	1	1	1	1	1	1	1	0	0	0	0
g	1	1	1	0	1	1	1	1	0	0	0	1
F_1	-	-	-	0	0	1	1	1	0	0	0	-
F_2	1	1	1	0	1	1	1	1	0	0	0	0

The segment verification method illustrated by this example requires that the circuit be segmented into subcircuits that each have fewer segment inputs that the number of circuit inputs. If the numbers of segment inputs are a,b,...c, the test length will be less than or equal to $2^a + 2^b + ... + 2^c$. The segmentation is best provided by the circuit designer. The process of circuit segmentation is similar to the system partitioning that must be done to fit a system on individual boards and chips. If the designer does not provide a circuit segmentation, the segments must be found by running a segmentation program, [Shperling 87], [Min 86], [Roberts 84].

In order to exhaustively test each subcircuit, all subcircuit inputs must be controllable at the input of the circuit and all subcircuit outputs must be observable at the circuit outputs. This can be achieved in two ways: (1) hardware partitioning and (2) sensitized partitioning. Access to the embedded inputs and outputs of the subcircuit under test can be achieved by inserting multiplexers and connecting the embedded inputs and outputs of each subcircuit to those primary inputs and outputs that are not used by the subcircuit under test [Bozorgui 80]. By controlling the multiplexers, all the inputs and outputs of each subcircuit can be accessed using primary input and output lines. Multiplexers can reduce the operating speed of the circuit and are costly to implement. However, it is possible to achieve the same testing discipline without actually inserting any multiplexers at all.

Circuit segmentation and subcircuit isolation can also be implemented by applying the appropriate input pattern to some of the input lines. The effect achieved is similar to that of hardware partitioning: paths from the primary inputs to the subcircuit inputs and paths from the subcircuit output to the primary output can be sensitized. Using these paths each subcircuit can be tested exhaustively. The methods to determine the appropriate inputs to sensitize the appropriate paths are based on the sensitization techniques of the well-known D Algorithm. The details are discussed in [Udell 86]. An example of the use of segment verification for the '181ALU is described in [McCluskey 81].

FAULT COVERAGE

The various test techniques described differ in their fault coverage. Exhaustive testing has the best coverage — all testable combinational faults. It is easy to show that output verification testing detects all testable combinational faults that do not increase the dependency of outputs on inputs (add entries to the dependence matrix). Segment verification testing is guaranteed to detect all testable combinational faults that are confined to a single segment, [Archambeau 84]. Faults that are not guaranteed to be detected may also be detected, but their detection cannot be guaranteed in general. In fact, it is not easy to generate examples of faults that are not detected by any of these test procedures. Such an example is given in [Archambeau 84], but it involves a circuit designed specifically to exhibit such a fault.

Single-stuck faults are guaranteed to be detected by exhaustive and pseudo-exhaustive tests so that fault simulation is not required to demonstrate 100% single-stuck fault coverage. Large segments are preferred for segment verification because they guarantee greater fault coverage than smaller segments.

BIST IMPLEMENTATIONS

There are two general classes of BIST architectures: (1) *external BIST*, in which test pattern generation and response analysis is done by circuitry that is separate from the functional circuitry being tested, and (2) *internal BIST*, in which the functional bistables are converted into test pattern generators and response analyzers, [Bardell 83], [McCluskey 85]. Some external BIST schemes test sequential logic directly by applying test patterns at the inputs and analyzing the responses, [Resnick 83], [Lake 86], [Vacca 87]. The techniques discussed here do not apply to such situations since the test patterns are for combinational circuits only. The BIST schemes discussed here all assume that the functional bistables of the circuit being tested can be converted into a scan path for testing, [McCluskey 86]. Such schemes are much more common than those that involve sequential circuit testing.

Only the test pattern generation circuitry is discussed here since signature analysis, [McCluskey 86], can be used to analyze the output response for any of the test schemes considered. Signature analysis is the only BIST output response analysis scheme currently in use.

For exhaustive testing, a full cycle counter is adequate for either external or internal BIST. A binary counter can be used, but a linear feedback shift counter modified to product the all-0 state is usually more economical, [McCluskey 86], [Wang 86C,D,E]. The modified LFSR is adequate since the sequence in which the patterns are applied is not critical. The implementation of exhaustive testing in the Intel 80386 is described in [Gelsinger 86] and [Gelsinger 87].

Many different schemes have been proposed for output verification testing. The simplest uses a full period counter to provide one test signal for each D(N) partition, [Barzilai 81], [McCluskey 83]. Multiplexers must be used to connect the same test signal to all inputs of a partition during testing. For non-MTC circuits in which more test signals are required than p, the number of partitions, the test length can often be reduced by using a verification test set to avoid the generation of all 2^p input patterns. Universal verification test sets can be generated with constant-weight counters, [Ichikawa 82]. The drawbacks to this technique are the difficulty of designing constant-weight counters and their high component count. These drawbacks are avoided at the expense of longer test times by using LFSRs to generate the VTS, verification test sets.

A simple scheme that uses maximum-length LFSRs to generate the VTS has an LFSR and a network of XOR gates to generate the test signals, [Akers 85], [Vasanthavada 85]. This approach is particularly efficient for circuits with low values of w (the number of inputs upon which each output depends). They are good for internal rather than external BIST architectures since they have fewer bistables than test signals, making it difficult to do the parallel-to-serial conversion required for shifting signals into the scan path.

It is also possible to use a non-maximum-length LFSR to generate the test patterns. Designs based on this structure require fewer components than those using full-length LFSRs, have the same length test sets for large w, and can be used for either internal or external BIST, but they are more complex to design than those using maximum-length LFSRs and an XOR network. The designs are based on using a non-primitive polynomial for the LFSR. The appropriate polynomial can be found by using methods based on coding theory. Techniques based on linear codes are described in [Wang 84], [Tang 84], [Wang 86A,B]. Other techniques are based on cyclic codes, [Wang 84B], [Wang 87A], shortened cyclic codes, [Wang 87B], and punctured cyclic codes, [Chen 87].

Less attention has been devoted to BIST implementations for segment verification tests. A design of such a structure for the 181 ALU is presented in [McCluskey 81]. In [Udell 87] general BIST methods for implementing segment verification are described.

CONCLUSIONS

Techniques for exhaustive and pseudo-exhaustive testing of combinational circuits have been presented. They are suitable for BIST. Compared to pseudo-random, testing they require more analysis in order to design the test pattern generation circuitry. The exhaustive techniques have the advantage of guaranteeing 100% detection of all testable single-stuck faults without the fault simulation needed to determine coverage of random tests. Exhaustive and pseudo-exhaustive tests provide very high coverage of failure modes that are not adequately represented as single-stuck faults such as multiple-stuck faults, bridging faults, intermittent failures. Of all the presently known test pattern generation methods, exhaustive and pseudo-exhaustive tests provide the highest quality test. Further research is needed to develop better structures for implementing pseudo-exhaustive testing.

EXHAUSTIVE AND PSEUDO-EXHAUSTIVE TESTING

REFERENCES

[Akers 85] Akers, S.B., "On the Use of Linear Sums in Exhaustive Testing," *Dig., Int'l Symp. on Fault-Tolerant Computing*, June 1985.

[Archambeau 84] Archambeau, E.C. and E.J. McCluskey, "Fault Coverage of Pseudo Exhaustive Testing," *Dig., 14th IEEE Int'l Conf. on Fault Tolerant Computing*, Orlando (Kissimmee) Florida, June 20-22, 1984.

[Archambeau 85] Archambeau, E.C., "Network Segmentation for Pseduo-Exhaustive Testing," CRC TR 85-10, July 1985.

[Bardell 83] Bardell, P.H. and W.H.McAnney, "Self-testing of Multichip Logic Modules," *Test and Measurement World*, pp. 26-29, March 1983.

[Barzilai 81] Barzilai, Z., J. Savir, G. Markowsky and M. Smith, "The Weighted Syndrome Sums Approach to VLSI Testing," *IEEE Trans. Comp.*, Vol. C-30, No. 12, pp. 996-1000, Dec. 1981.

[Bozorgui-Nesbat 80] Bozorgui-Nesbat, S., and E.J. McCluskey, "Structured Design for Testability to Eliminate Test Pattern Generation," *FTCS-10*, pp. 158-163, Oct. 1980.

[Camenzind 72] Camenzind, H.R., "Testability," Chap. 20 in *Electronic Integrated Systems Design*, Van Nostrand Reinhold, NY, 1972.

[Chandra 83] Chandra, A.K., et al., "On Sets of Boolean n-Vectors with all k-Projections Subjective," *Acta Informatica*, Vol. 20, pp. 103-111, 1983.

[Chen 87] Chen, C.L., "Exhaustive Test Pattern Generation Using Cyclic Codes," *IEEE Trans. Comp.*, Nov. 1987

[Chrones 81] Chrones, D. "Parametric Testers Evaluate Wafer Processing," *EDN*, pp. 117-122, Apr. 15, 1981.

[Doyle] Doyle, E., Jr. and B. Morris, "Microelectronic Failure Analysis Techniques, A Procedural Guide," *G.E. Co. and RADC.*

[Eldred 59] Eldred, R.D., "Test Routines Based on Symbolic Logical Statements," *J. ACM*, Vol. 6, No. 1, pp. 33-36, 1959.

[Franz 85] Franz, M., "Input Vectors Drive Simulation of Logic-Array Designs," *EDN*, pp. 153-158, June 13, 1985.

[Friedman 71] Friedman, A.D., and P.R. Menon, *Fault Detection in Digital Circuits*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[Gelsinger 86] Gelsinger, P., "Built In Self Test of the 80386," *Proc. Int'l Conf. Computer Design*, pp. 169-173, Oct. 1986.

[Gelsinger 87] Gelsinger, P., "Design and Test of the 80386," *IEEE Design & Test, pp.* 42-50, June 1987.

[Ichikawa 82] Ichikawa, M., "Constant Weight Code Generators," CRC TR 82-7.

[Jha 85] Jha, N.K., and J.A. Abraham, "Design of Testable CMOS Logic Circuits under Arbitrary Delays," *IEEE Trans. Computer-Aided Design*, Vol. CAD-4, No. 3, pp. 264-269, July 1985.

[Koeppe 87] Koeppe,S. "Layout Methods to Reduce CMOS Stuck-Open-Faults and Enhance Testability," *ISSCC 87*, pp. 228-229, Feb. 26, 1987.

[Lake 86] Lake, R., "A Fast 20K Gate Array with On-Chip Test System," *VLSI Systems Design*, pp. 46-55, June 1986.

[Liu 87] Liu, W.L., and E.J.McCluskey, "CMOS Scan-path IC Design for Stuck-open Fault Testability," CRC TR 87-10, June 1987; to appear in *IEEE Jrn'l. Solid-State Circuits*.

[McCluskey 80] McCluskey, E.J., and S. Bozorgui-Nesbat, "Design for Autonomous Test," *Proc., 1980 IEEE Test Conf.*, pp. 15-21, Nov. 1980.

[McCluskey 81] McCluskey, E.J. and S. Bozorgui-Nesbat, "Design for Autonomous Test," *IEEE Trans. Comp.*, pp. 866-875, Nov. 1981;. also CRC TR 81-1.

[McCluskey 82A] McCluskey, E.J., "Verification Testing," *Dig., 19th Ann. Design Automation Conf.*, Las Vegas, Nev., pp. 495-500, June 14-16, 1982.

[McCluskey 82B] McCluskey, E.J., "Built-in Verification Test," *Dig., 1982 IEEE Test Conf.*, Nov. 11-13, 1982, pp. 183-190.

[McCluskey 83] McCluskey, E.J., "Exhaustive and Pseudo-exhaustive Test," Built-in Test — Concepts and Techniques, Tutorial, *ITC83*.

[McCluskey 84A] McCluskey, E.J., "Pseudo-Exhaustive Testing for VLSI Devices," *ATE Silicon Valley Conf.*, San Mateo, CA, pp. IV-5 — IV-21, Apr. 10-12, 1984.

[McCluskey 84B] McCluskey, E.J., "Verification Testing — A Pseudoexhaustive Test Technique," *IEEE Trans. Comp.*, pp. 541-546.

[McCluskey 84C] McCluskey, E.J., "VLSI Design for Testability," *1984 Symposium on VLSI Technology*, San Diego, CA, Sept. 10-12, 1984.

[McCluskey 84D] McCluskey, E.J., "A Survey of Design for Testability Scan Techniques," *VLSI Systems Design*, Vol. V, No. 12, pp. 38-61, Dec. 1984.

[McCluskey 85] McCluskey, E.J., "Built-In Self-Test Structures," *IEEE Design & Test of Computers*, pp. 29-36, Apr. 1985.

[McCluskey 86] McCluskey, E.J., "Logic Design Principles: With Emphasis on Testable Semicustom Circuits," *Prentice-Hall*, Englewood Cliffs, NJ, 1986.

[Mei 74] Mei, K.C.Y., "Bridging and Stuck-at Faults," *IEEE Trans. Comput.* , Vol. C-23, No. 7, pp. 720-727, July 1974.

[Min 86] Min, Y., and Z. Ki, "Pseudo-Exhaustive Testing Strategy for Large Combinatorial Circuits," *Computer Systems Science and Engineering*, Vol. 1, No. 4, pp. 213-220, Oct. 1986.

[Moore 55] Moore, E.F., "Gedanken Experiments on Sequential Machines," in Automata Studies, C.E. Shannon and J. McCarthy, Eds., pp. 129-153.

[Reddy 83] Reddy, S.M., M.K. Reddy, and J.G. Kuhl, "On Testable Design for CMOS Logic Circuits," *Proc., International Test Conf.*, Philadelphia, PA, pp. 435-445, Oct. 18-20, 1983.

[Resnick 83] Resnick, D., "Testability and Maintainability with a New 6K Gate Array," *VLSI Systems Design*, Mar./Apr. 1983.

[Roberts 84] Roberts, M.W., and P.K. Lala, "An Algorithm for the Partition of Logic Circuits," *IEEE Proc.*, Vol. 131, Pt. E, No. 4, July 1984.

[Sphlerling 87] Spherling, I., and E.J. McCluskey, "Circuit Segmentation for Pseudo-Exhaustive Testing via Simulated Annealing," CRC 87-2, Jan. 1987; to appear in *ITC 87*.

[Stover 84] Stover, A.C., "ATE: Automatic Test Equipment," *Mc Graw-Hill Book Co.*, 1984.

[Tang 83] Tang, D.T., and L.S. Woo, "Exhaustive Test Pattern Generation with Constant Weight Vectors," *IEEE Trans. Comp.*, Vol. C-32, No. 12, pp. 1145-1150, Dec. 1983.

[Tang 84] Tang, D.T., and C.L. Chen, "Logic Test Pattern Generation Using Linear Codes," *IEEE Trans. Comput.*, Vol. C-33, No. 9, pp. 845-850, Sept. 1984.

[Turner 85] Turner, M.E., D.G. Leet, R.J. Prilik and D.J. McLean, "Testing CMOS VLSI: Tools, Concepts, and Experimental Results," *Int'l Test Conf.*, Philadelphia, PA, pp. 322-328, Nov. 19-21, 1985.

[Udell 86] Udell, J.G., Jr., "Test Set Generation for Pseudo-Exhaustive BIST," *Digest of Technical Papers, 1986 IEEE International Conference on Computer-Aided Design*, Santa Clara, CA, pp. 52-55, Nov. 10-13, 1986.

[Udell 87] Udell, J.G., Jr., "Reconfigurable Hardware for Pseudo-Exhaustive Test," CRC TR 87-4, Feb. 1987.

[Vacca 87] Vacca, Tony, et al., "A Cryogenically Cooled CMOS VLSI Supercomputer," *VLSI Systems Design*, pp. 80-88, June 1987.

[Vasanthavada] Vasantharada, N., and P.N. Marinos, "An Operationally Efficient Scheme for Exhaustive Test-Pattern Generation Using Linear Codes," *Proc., IEEE 1985 Int'l Test Conference*, pp. 476-482, Durham, NC, 1985.

[Wadsack 78] Wadack, R.L., "Fault Modelling and Logic Simulation of CMOS and MOS Integrated Circuits," *BSTJ*, Vol. 57, No. 5, pp. 1449-1473, May-June 1978.

[Wang 84] Wang, L.-T., and E.J. McCluskey, "A New Condensed Linear Feedback Shift Register Design for VLSI System Testing," *FTCS-14*, pp. 360-365, June 20-22, 1984.

[Wang 86A] Wang, L.-T., and E.J. McCluskey, "Condensed Linear Feedback Shift Register (LFSR) Testing — A Pseudoexhaustive Test Technique," Special Issue on Fault-Tolerant Computing, *IEEE Trans. Comput.*, Vol. C-35, No. 4, pp. 367-370, Apr. 1986.

[Wang 86B] Wang, L.-T., and E.J. McCluskey, "Circuits for Pseudo-Exhaustive Test Pattern Generation," *Proc., IEEE 1986 Int'l Test Conference*, Washington, DC, pp. 25-37, Sept. 8-10, 1986.

[Wang 86C] Wang, L.-T., and E.J. McCluskey, "A Hybrid Design of Maximum-Length Sequence Generators," *Proc., IEEE 1986 Int'l Test Conference*, Washington, DC, pp. 38-47, Sept. 8-10, 1986.

[Wang 86D] Wang, L.-T., and E.J. McCluskey, "Complete Feedback Shift Register Design for Built-In Self-Test," *Digest of Papers, IEEE 1986 Int'l Conference on Computer-Aided Design*, pp. 56-59, Santa Clara, CA, Nov. 11-13, 1986.

[Wang 86E] Wang, L.-T., and E.J. McCluskey, "Feedback Shift Registers for Self-Testing Circuits," *VLSI Systems Design*, pp. 50-55, Dec. 1986.

[Wang 87A] Wang, L.-T., and E.J. McCluskey, "Linear Feedback shfit Register Design Using Cyclic Codes"; to appear in *IEEE Trans. Comput.*, 1987.

[Wang 87b] Wang, L.-T., and E.J. McCluskey, "Circuits for Pseudo-Exhaustive Test Pattern Generation Using Shortened Cyclic Codes," to appear in *IEEE 1987 Int'l Conference on Computer Design: VLSI In Computers & Processors*, Port Chester, NY, Oct. 5-8, 1987.

[Williams 85] Williams, T.W., "Test Length in a Self-testing Environment," *IEEE Design and Test*, pp. 59-63, Apr. 1985.

[Zasio 85] Zasio, J.J., "Non-stuck Fault testing of CMOS VLSI," *Proc., COMPCON Spring 85*, San Francisco, CA, pp. 388-391, Feb. 26-28.

CRC TR = Stanford Center for Reliable Computing, Technical Report

EXHAUSTIVE AND PSEUDO–EXHAUSTIVE TEST

E.J. McCluskey

CENTER for RELIABLE COMPUTING

Computer Systems Laboratory
Departments of Computer Science
and Electrical Engineering

Stanford University
Stanford, CA 94305-4055

EXHAUSTIVE AND PSEUDO–EXHAUSTIVE TEST

- Introduction
- Exhaustive Test
- Pseudo-exhaustive Test
  - Output Function Verification
  - Segment Function Verification

TEST INPUT PATTERN SELECTION

Fault Model Based

Function Based

Random and Pseudo-random

Exhaustive Test and Pseudo-exhaustive

(All assume faults do not increase number of internal states)

EXHAUSTIVE and PSEUDO-EXHAUSTIVE TEST

ADVANTAGES

No ATPG program required

No fault simulation required

No circuit modification required
to increase fault coverage

Very high fault coverage

Suitable for built-in self test

EXHAUSTIVE TEST DISADVANTAGES

Long Test Time

Large Volume of Test Output Data

Combinational circuits require 2^n test vectors for n-input circuit

Sequential circuits require $<0(2^n)(k^2)$ test vectors for k–state circuit

EXHAUSTIVE TEST

CIRCUITS TO GENERATE INPUT PATTERNS

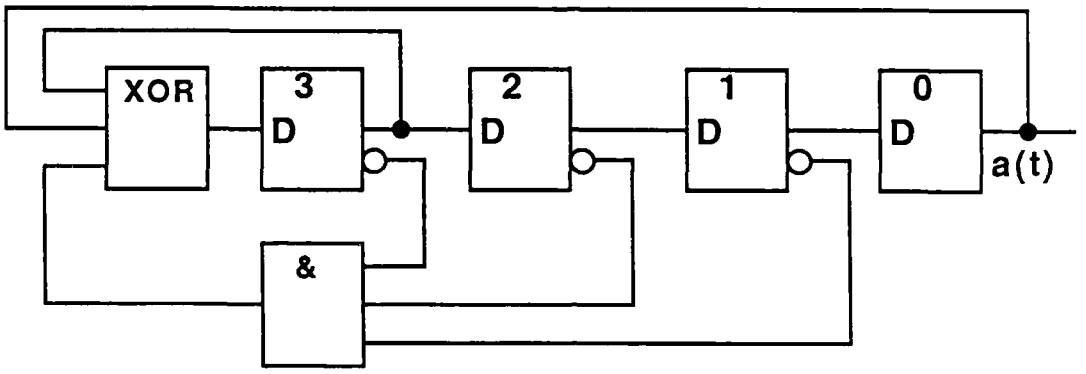
Binary Counter

Linear Feedback shift register

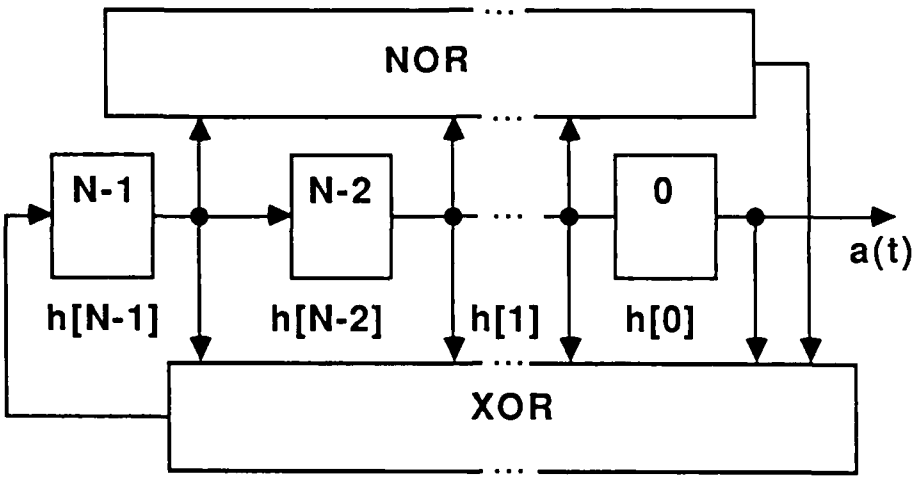
1	0	0	0		1	1	0	1
1	1	0	0		0	1	1	0
1	1	1	0		0	0	1	1
1	1	1	1		1	0	0	1
0	1	1	1		0	1	0	0
1	0	1	1		0	0	1	0
0	1	0	1		0	0	0	1
1	0	1	0		(0	0	0	0)

EXHAUSTIVE TEST

4-STAGE LFSR MODIFIED FOR 16 STATES

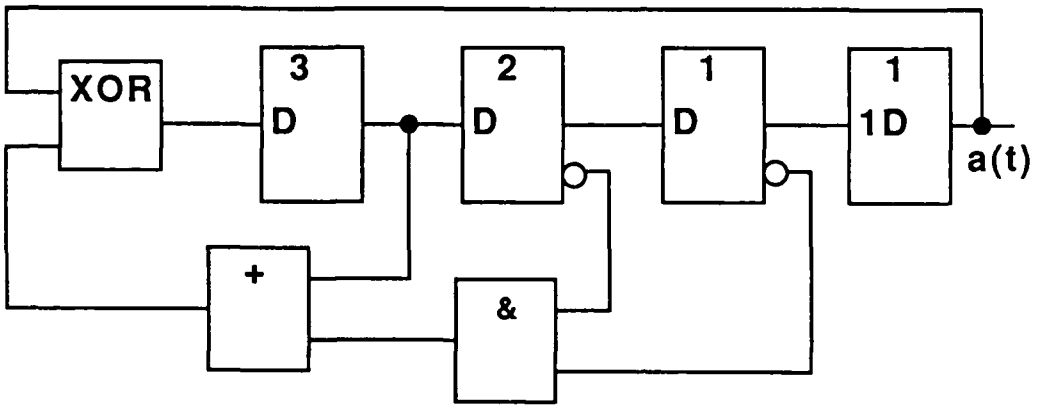


N-STAGE LFSR MODIFIED FOR 2^N STATES



EXHAUSTIVE TEST

EFFICIENT 16 STATE LFSR



PSEUDO–EXHAUSTIVE TEST

USED WHEN EXHAUSTIVE TEST TOO LONG

Individual Output Verification
- Exhaustive Test of Each OUtput
- No Output Depends on All Inputs

Network Partitioned or Segmented
- Exhaustive Test of Each Segment

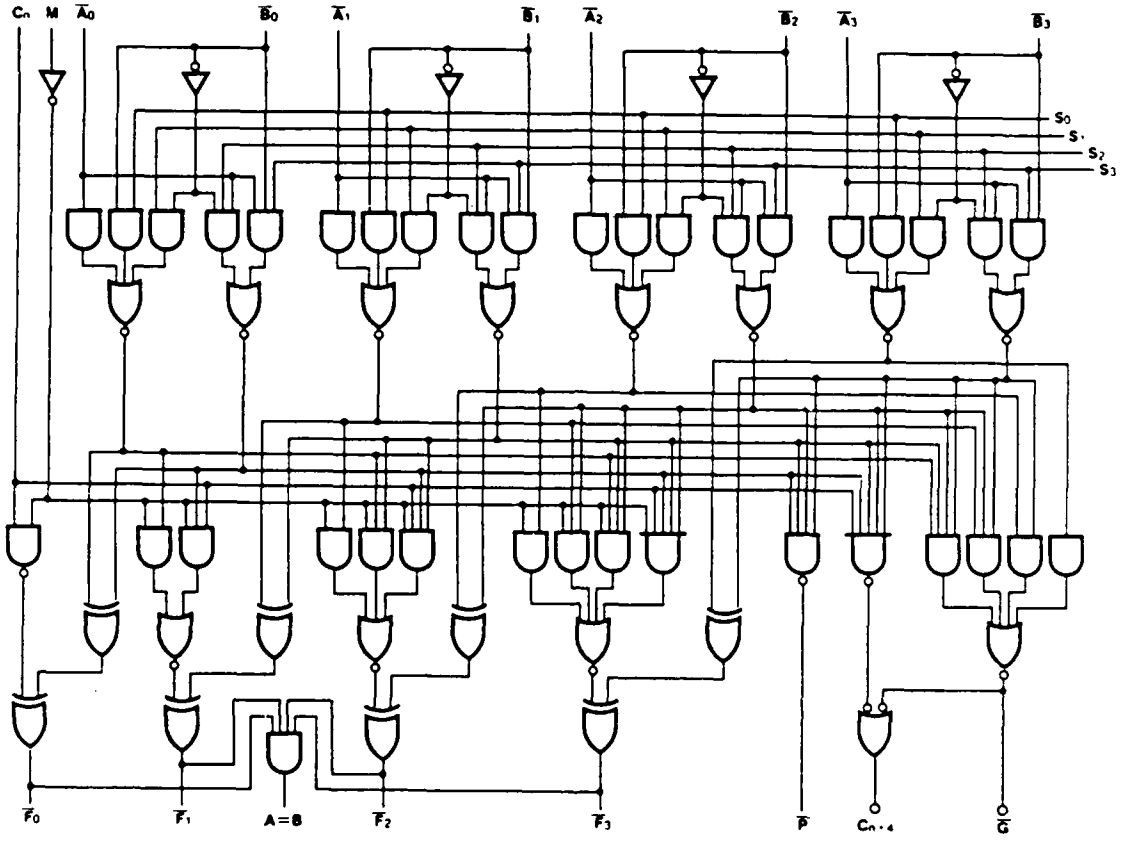
COMBINATIONAL CIRCUIT CLASSIFICATION

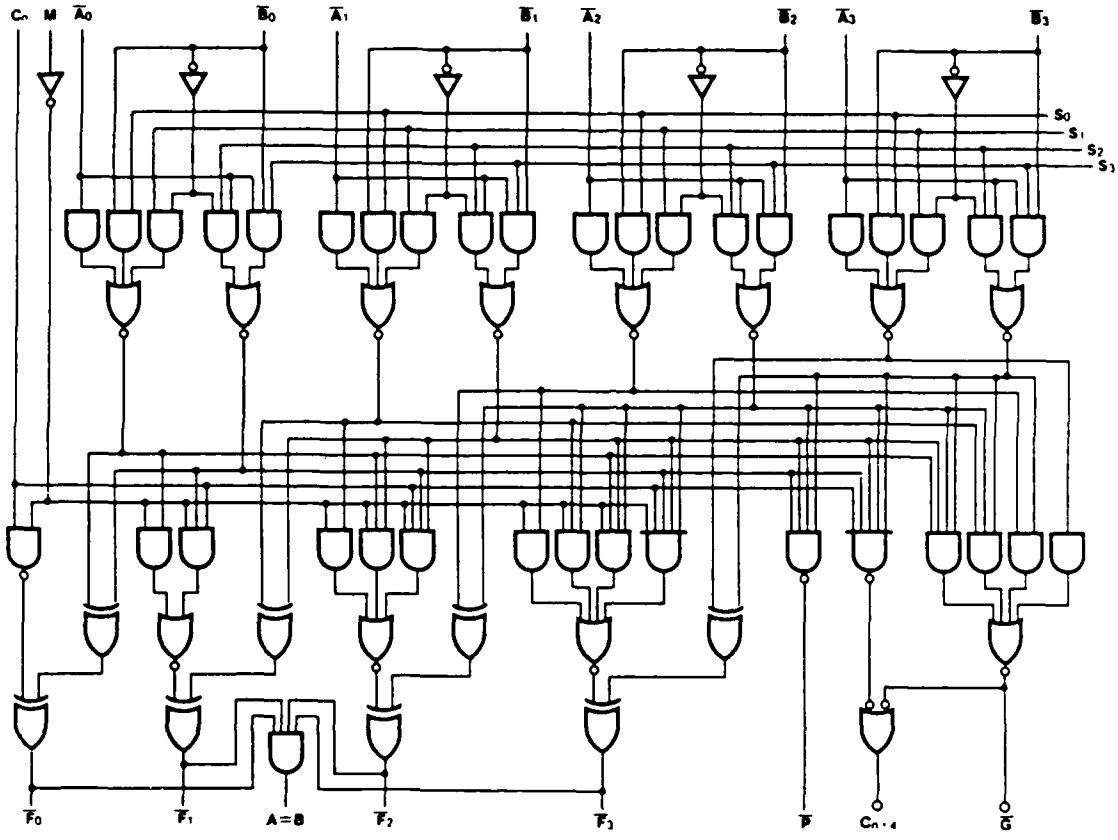
PARTIAL DEPENDENCE CIRCUIT - PDC
No output depends on all inputs

Verification test techniques useful

FULL DEPENDENCE CIRCUIT - FDC
Some output depends on all inputs

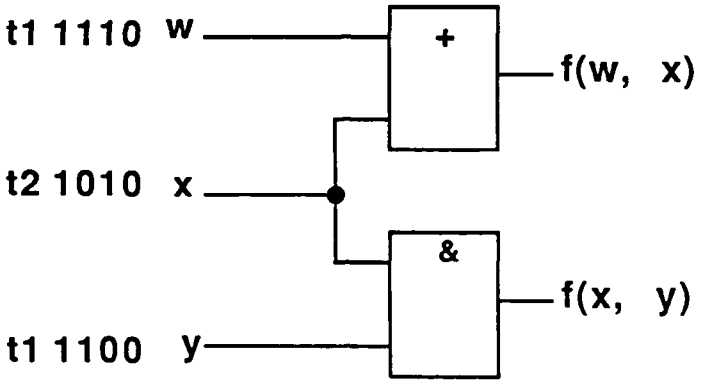
Segmenting (partitioning) required
for pseudo-exhaustive testing


Cn
M
A0
B0
A1
B1
A2
B2
A3
B3
S0
S1
S2
S3
F0
F1
A=B
F2
F3
P
Cn+4
G


Cn
M
A0
B0
A1
B1
A2
B2
A3
B3
S0
S1
S2
S3
F0
F1
A=B
F2
F3
P
Cn+4
G

PSEUDO - EXHAUSTIVE TEST
INDIVIDUAL OUTPUT VERIFICATION

Serial vs. Parallel



Two test signals
Maximal Test Concurrency (MTC)
Circuit

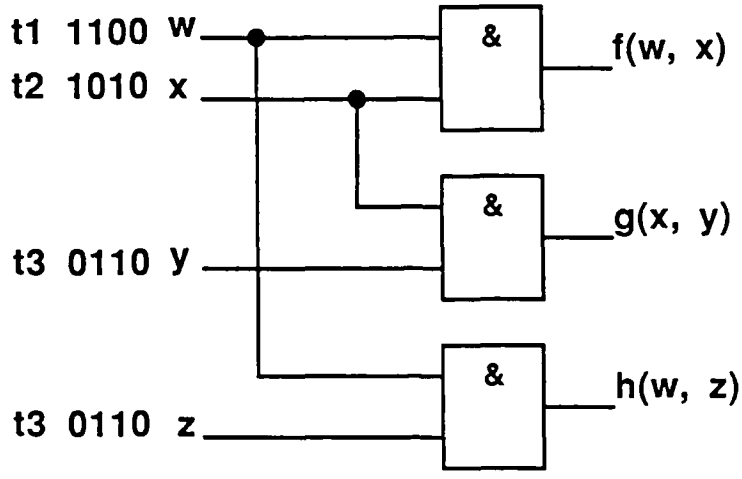
PDC CLASSIFICATION

MAXIMAL TEST CONCURRENCY CIRCUIT
(MTC Circuit)

Number of test signals required = Maximum number of inputs connected to any output

NON-MAXIMAL TEST CONCURRENCY CIRCUIT
(NMTC Circuit)

Number of test signals required = Greater than the maximum number of inputs connected to any output



Four - input NMTC circuit
3 test signals – 4 test patterns
minimum - length test

Circuit of example 2

D(N)

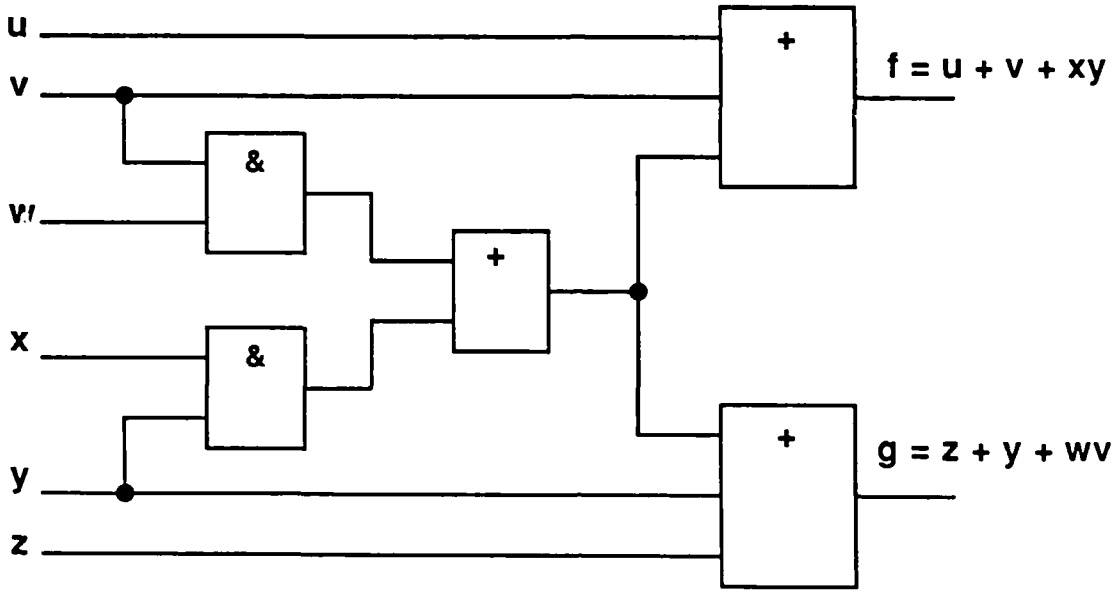
outputs	inputs w	x	y	z
f (w, x)	1	1	0	0
g (x, y)	0	1	1	0
h (w, z)	1	0	0	1

DP(N)

outputs	inputs w	x	y	z
f (w, x)	1	1	0	0
g (w, y)	0	1	1	0
h (x, z)	1	0	0	1

Each row of a partition can have at most 1 entry

DP(N) is not Unique
wy or xz partition possible



$I_f = \{u, v, w, x, y\}$ $I_g = \{v, w, x, y, z\}$

Network to illustrate calculation of I_j

outputs	inputs w	x	y z
f (w, x)	1	1	0
g (w, y)	1	0	1
h (x, z)	0	1	1

RDP(N)

	t1	t2	t3
test patterns	1	1	0
	1	0	1
	0	1	1
	0	0	0

RVTS(N)

NMTC Circuit

Sharp VTS – 4 test patterns

VERIFICATION TEST SETS - VTS

Each output has its inputs cycled through all possible combinations

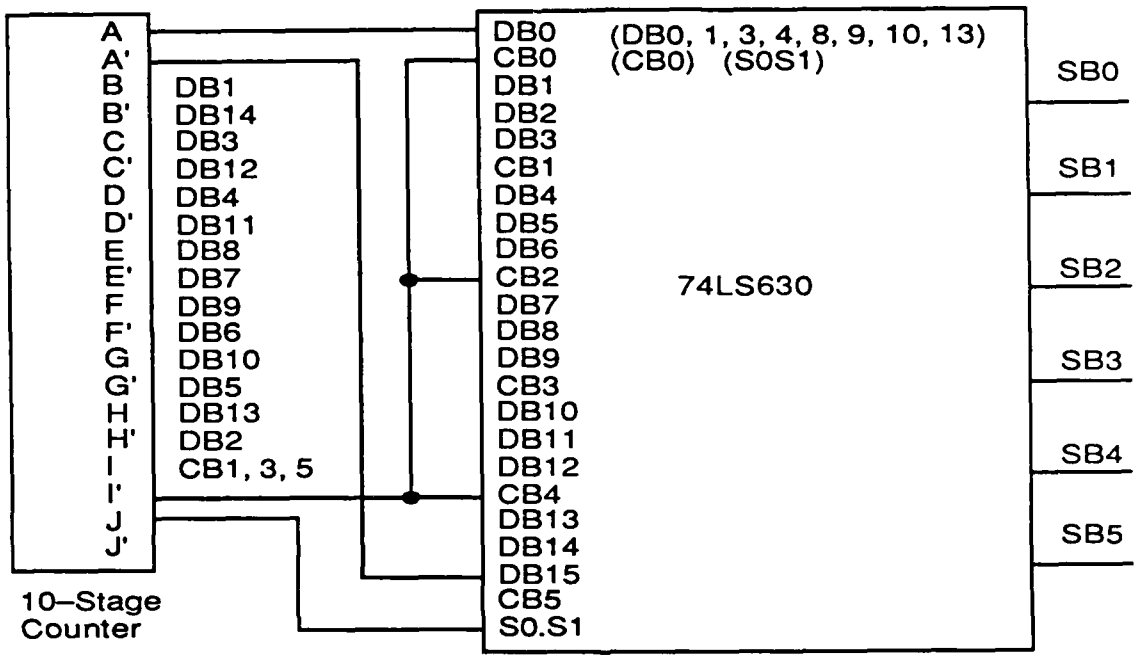
SHARP VERIFICATION TEST SET

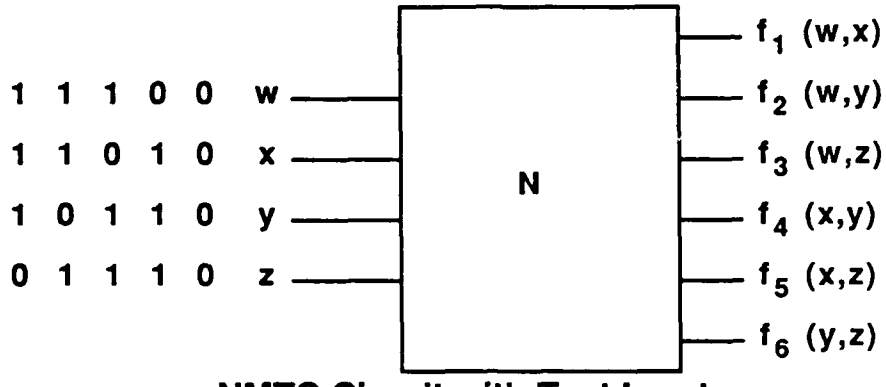
Test Set Length = Maximum length of exhaustive test set for single output

Minimum length VTS
Meets lower bounds on VTS
Always exists for MTC circuit
Sometimes exists for NMTC circuit

D(N) for 74S630 Network

	DATA BITS																CHECK BITS						
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	S0.S
SB0	1	1	0	1	1	0	0	0	1	1	1	0	0	1	0	0	1	0	0	0	0	0	1
SB1	1	0	1	1	0	1	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	1
SB2	0	1	1	0	1	1	0	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1
SB3	1	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	1	0	0	1
SB4	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1
SB5	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1


A
A'
B
B'
C
C'
D
D'
E
E'
F
F'
G
G'
H
H'
I
I'
J
J'
DB1
DB14
DB3
DB12
DB4
DB11
DB8
DB7
DB9
DB6
DB10
DB5
DB13
DB2
CB1, 3, 5
10–Stage
Counter
DB0
CB0
DB1
DB2
DB3
CB1
DB4
DB5
DB6
CB2
DB7
DB8
DB9
CB3
DB10
DB11
DB12
CB4
DB13
DB14
DB15
CB5
S0.S1
(DB0, 1, 3, 4, 8, 9, 10, 13)
(CB0) (S0S1)
74LS630
SB0
SB1
SB2
SB3
SB4
SB5



NMTC Circuit with Test Inputs
4-Inputs, 6-Outputs, 5 Test Patterns
No Sharp VTS Possible

$f_1(vwx)$	| 1 1 1 0 0 |	
$f_2(vwy)$	| 1 1 0 1 0 |	Dependence
$f_3(vxz)$	| 1 0 1 0 1 |	Matrix
$f_4(vyz)$	| 1 0 0 1 1 |	
$f_5(wxy)$	| 0 1 1 1 0 |	
$f_6(wxz)$	| 0 1 1 0 1 |	
$f_7(wyz)$	| 0 1 0 1 1 |	
$f_8(xyz)$	| 0 0 1 1 1 |	
	v w X Y z	
	| 1 0 0 0 0 |	
	| 0 0 0 1 1 |	Verification
	| 0 0 1 0 1 |	Test Set
Test	| 1 0 1 1 0 |	
Patterns	| 1 1 0 0 1 |	NMTC Circuit
	| 0 1 0 1 0 |	Sharp VTS
	| 0 1 1 0 0 |	8 Test Patterns
	| 1 1 1 1 1 |	

	U (5,3) v w x y z	
1	1 0 0 0 0	Universal
2	0 1 0 0 0	Verification
3	0 0 1 0 0	Test Set
4	0 0 0 1 0	for
5	0 0 0 0 1	5 Test Signals
6	0 1 1 1 1	(p = 5)
7	1 0 1 1 1	Maximum output
8	1 1 0 1 1	Dependence on
9	1 1 1 0 1	3 Inputs
10	1 1 1 1 0	(w = 3)

Sharp VTS not possible

Sharp Verification Test Set for
p = number of test signals = 3
w = Maximum Output Dependence = 2
(p = w + 1)

Even Parity			Odd Parity		
x	y	z	x	y	z
0	0	0	0	0	1
0	1	1	0	1	0
1	0	1	1	0	0
1	1	0	1	1	1

Sharp VTS always possible for p = W + 1
using all constant parity vectors

Specifications for U(p, w)

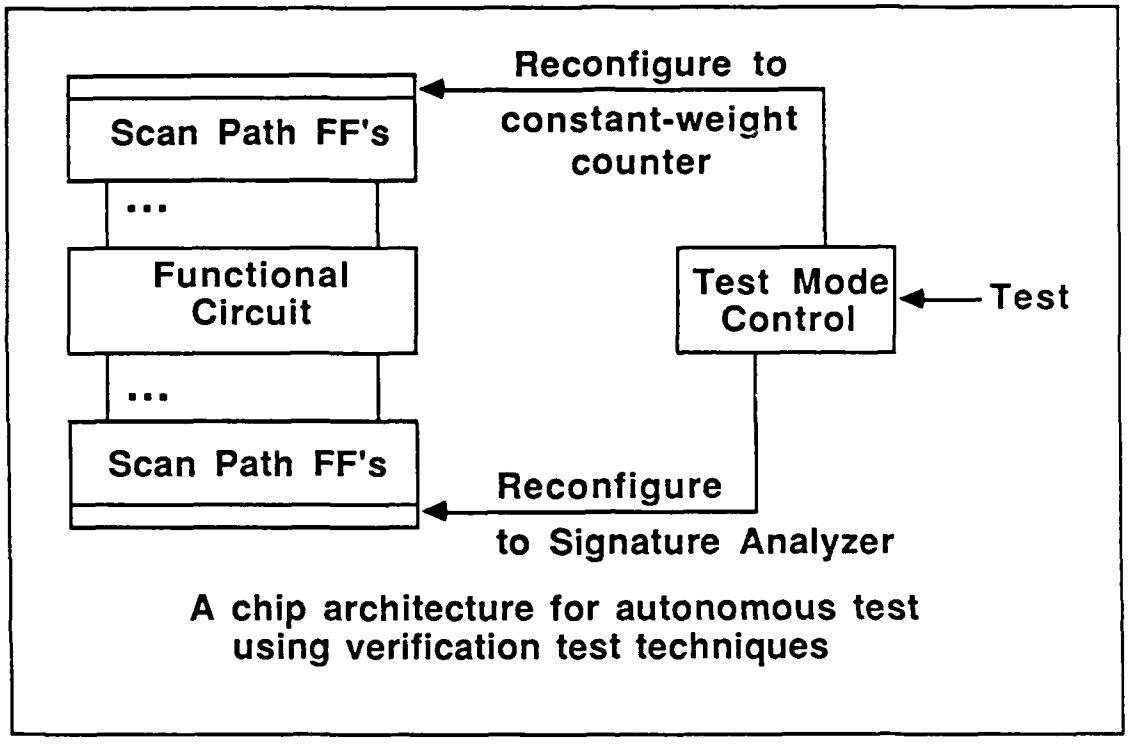
w	U(p,w)		Range	Number of Tests (rows)
2	p[0, p–1]	p[1, p]	p > 3	p + 1
3	p[1, p–1]		p > 4	2p
4	p[1, p–2]	p[2, p–1]	p > 5	1/2 p(p + 1)
5	p[2, p–2]		p > 6	p(p – 1)
6	p[2, p–3]	p[3, p–2]	p > 8	B(p + 1, 3)
6	p[1, 4, 7]		p = 8	1/2 p(p + 1)
7	p[3, p–3]		p > 9	2 B(p, 3)
7	p[0, 3, 6, 9]		p = 9	170

Specifications for U(p, w)

w	U(p,w)		Range	Number of Tests (rows)
8	p[3, p–4]	p[4, p–3]	p > 11	B(p + 1, 4)
8	p[0, 3, 6, 9]	p[1, 4, 7, 10]	p = 10	341
8	p[0, 4, 8]	p[3, 7, 9]	p = 11	496
9	p[4, p–4]		p > 12	2 B(p, 4)
9	p[1, 4, 7, 10]		p = 11	682
9	p[0, 4, 8, 12]		p = 12	992
10		p[5, p–4]	p > 14	B(p + 1, 5)
10		p[2, 5, 8, 11]	p = 12	1365
10		p[1, 5, 9, 13]	p = 13	2016
10		p[4, 9, 14]	p = 14	3004

Comparison of Different Test Lengths for Worst-Case Circuit

Technique	Test Length		
	n = p, w	w = 10	n=23, w=10
Exhaustive Test	2^n	2^n	8,388,608.
Universal Verification Test	U(p,w)	B(n+1,5)	42,504.
Minimum Verification Test	———	———	1,024.



A chip architecture for autonomous test using verification test techniques

PSEUDO-EXHAUSTIVE TEST
INDIVIDUAL OUTPUT VERIFICATION

Fault Coverage

ALL Irredundant Faults that
leave UNCHANGED or DECREASE
dependency of Output Functions
on Input Variables

SOME Irredundant Faults that
INCREASE
dependency of Output Functions

All Irredundant Single Stuck Faults

PET OUTPUT FUNCTION VERIFICATION

Input Pattern Generation

Constant-weight Counter

Linear Feedback Shift Register

PET SEGMENT FUNCTION VERIFICATION

Select Network Segments

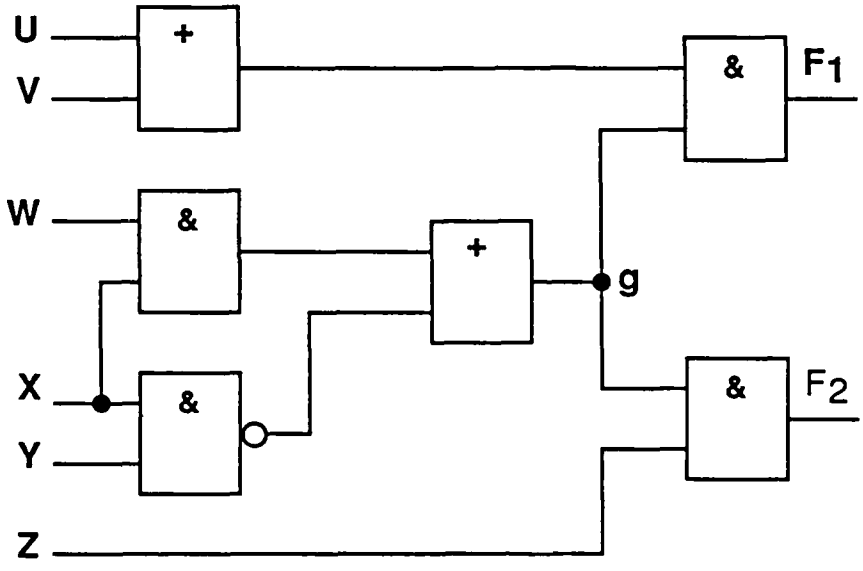
Test Time Order $2^{n_1}+\ldots+2^{n_k}$

instead of

Test Time Order $2^{n_1+\ldots+n_k}$

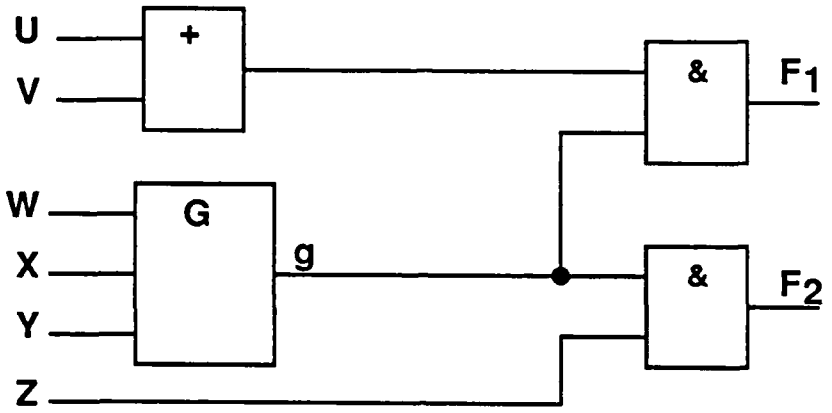
Implement Segments using
Path Sensitization or
Multiplexers

SIMPLE SEGMENT EXAMPLE

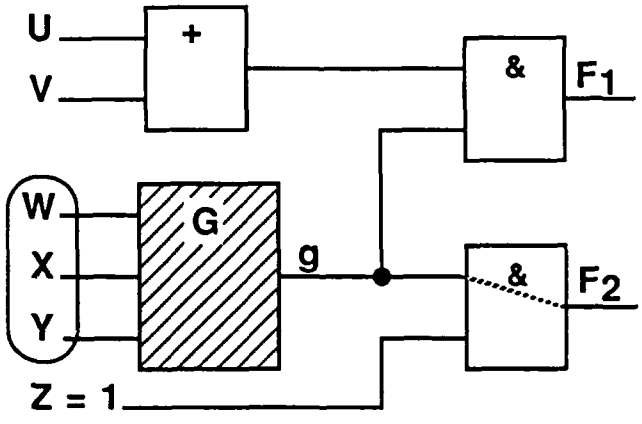


EXHAUSTIVE TEST LENGTH: 64

PET OUTPUT VERIFICATION LENGTH: 32

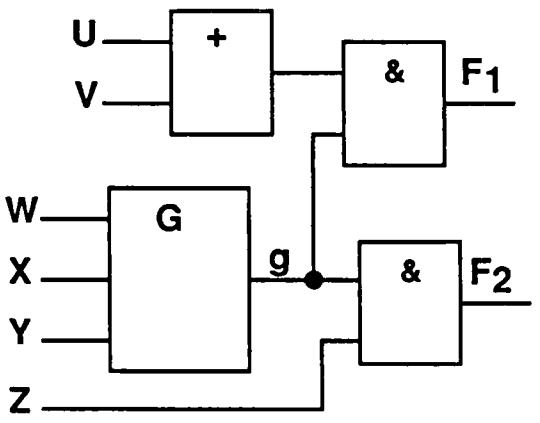


Segmentation of Simple Example Network



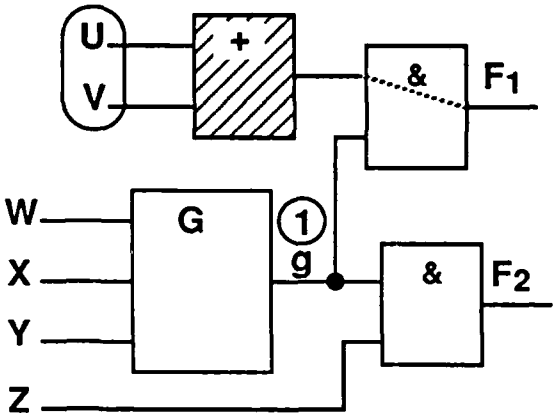
UV	WXY	Z	g	F_1	F_2
	000	1	1		1
	001	1	1		1
	010	1	1		1
	011	1	0		0
	100	1	1		1
	101	1	1		1
	110	1	1		1
	111	1	1		1

Exhaustive Test of g Sensitized to F_2 by Z = 1



UV	WXY	Z	g	F_1	F_2
	000	1	1		1
	001	1	1		1
	010	1	1		1
	011	1	0		0
	100	1	1		1
	101	1	1		1
	110	1	1		1
	111	1	1		1
	111	0	1		0
	011	0	0		0

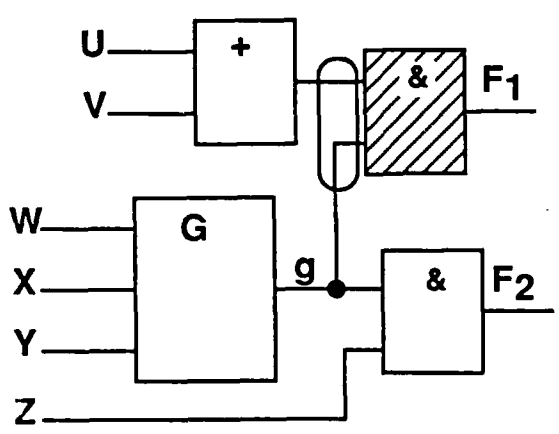
Exhaustive test of F_2 AND gate added to Exhaustive Test of G



UV	WXY	Z	g	F_1	F_2
	000	1	1		1
	001	1	1		1
	010	1	1		1
	011	1	0		0
00	100	1	1	0	1
01	101	1	1	1	1
10	110	1	1	1	1
11	111	1	1	1	1
	111	0	1		0
	011	0	0		0

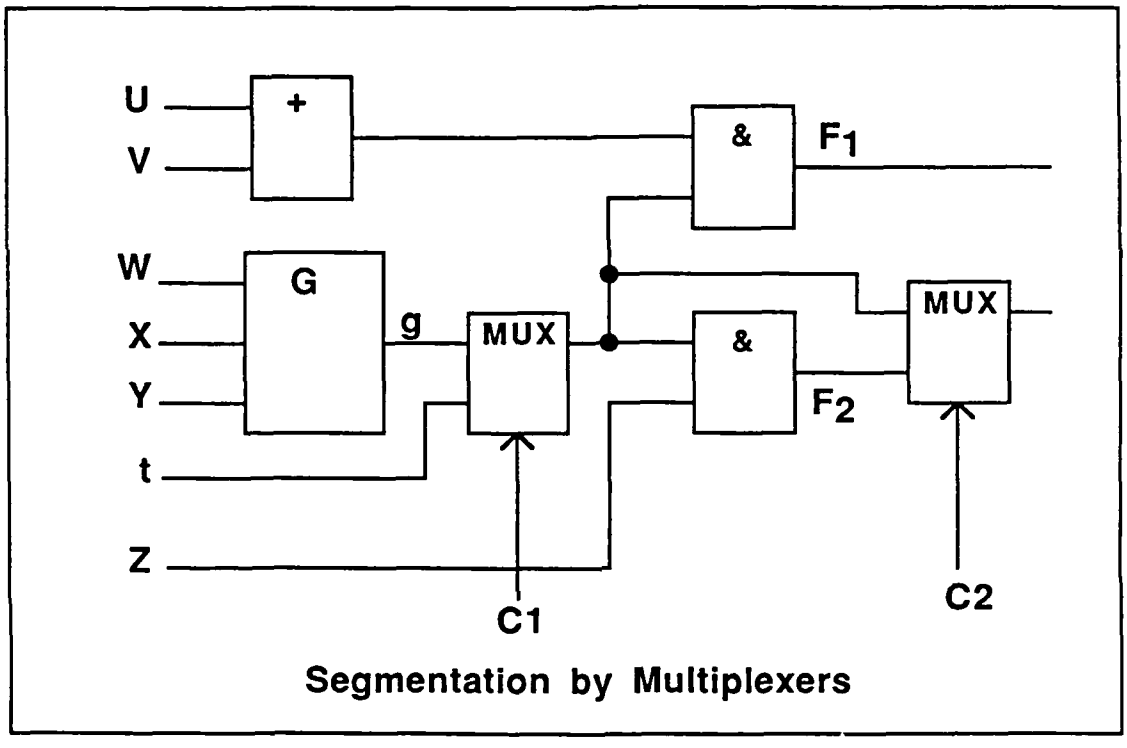
UV OR gate sensitized to F_1 by g = 1.

Exhaustive Test of F_1 OR gate
Added to Exhaustive Tests of F_2 AND gate and G



UV	WXY	Z	g	F_1	F_2
	000	1	1		1
	001	1	1		1
	010	1	1		1
00	011	1	0	0	0
00	100	1	1	0	1
01	101	1	1	1	1
10	110	1	1	1	1
11	111	1	1	1	1
	111	0	1		0
11	011	0	0	0	0

Exhaustive Test of F_1 AND gate added to Exhaustive Tests of F_1 OR gate, F_2 AND gate, G

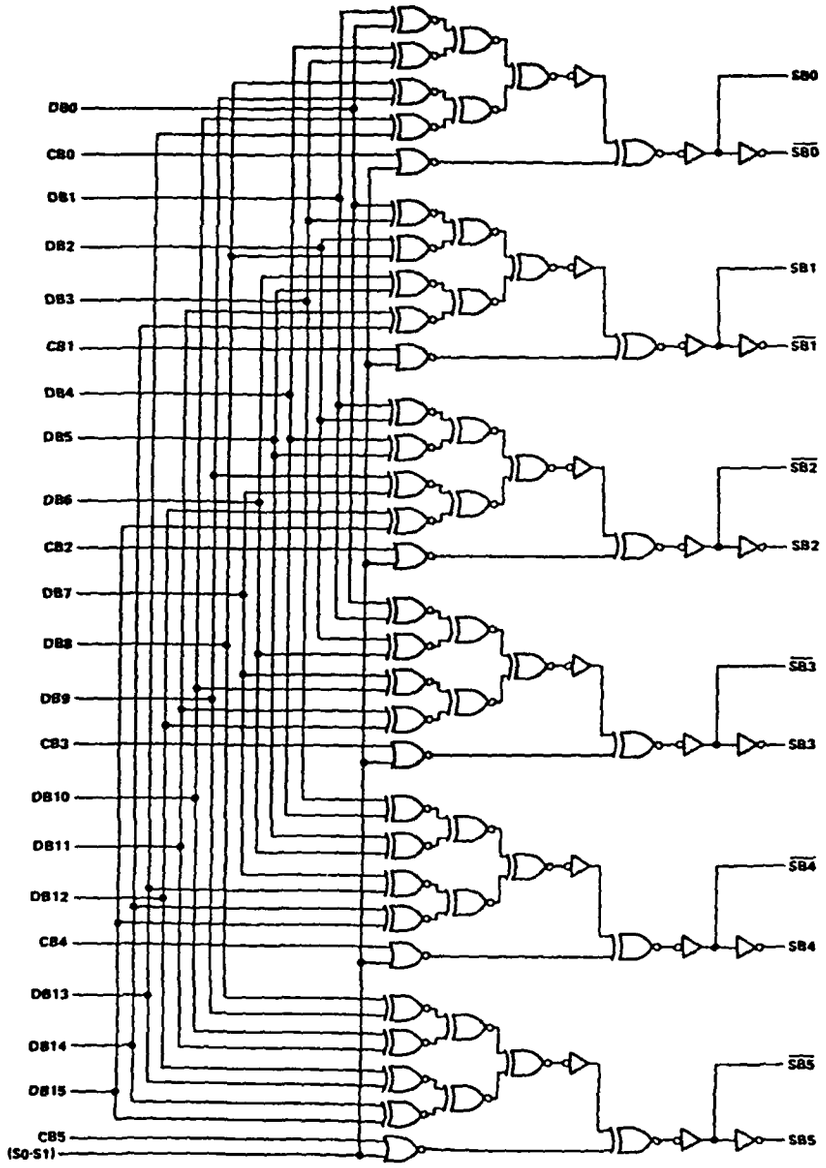


Segmentation by Multiplexers

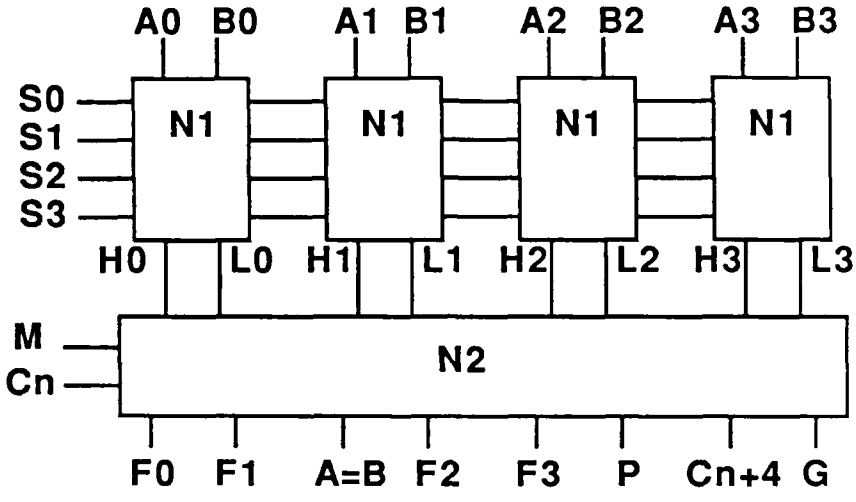
PET SEGMENT FUNCTION VERIFICATION

Simple Example Network Test Lengths

Exhaustive Test	64
PET Output Function Verification	32
PET Segment Function Verification	
Path Sensitization	10
Multiplexers	13

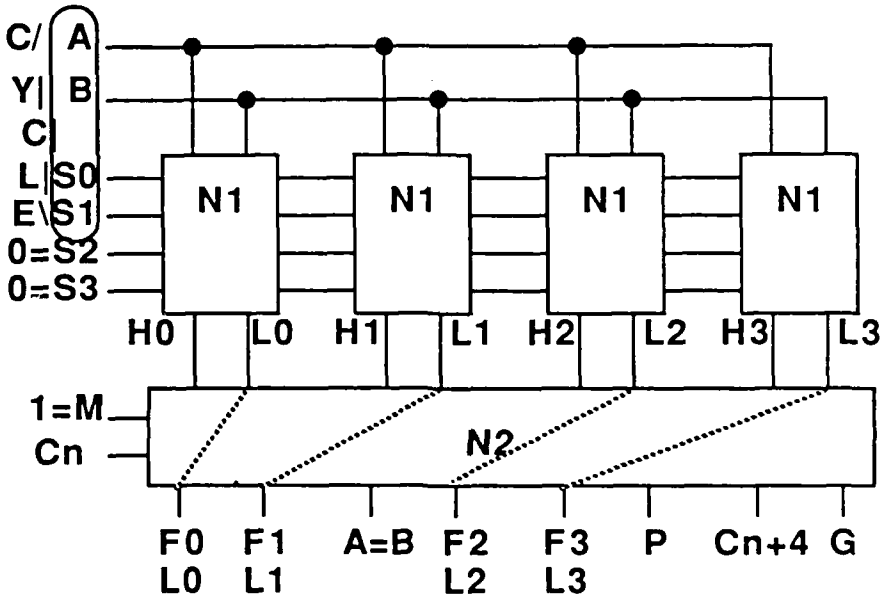

DB0
CB0
DB1
DB2
DB3
CB1
DB4
DB5
DB6
CB2
DB7
DB8
DB9
CB3
DB10
DB11
DB12
CB4
DB13
DB14
DB15
CB5
(S0-S1)
SB0
SB0
SB1
SB1
SB2
SB2
SB3
SB3
SB4
SB4
SB5
SB5

SEGMENTATION OF 181 ALU

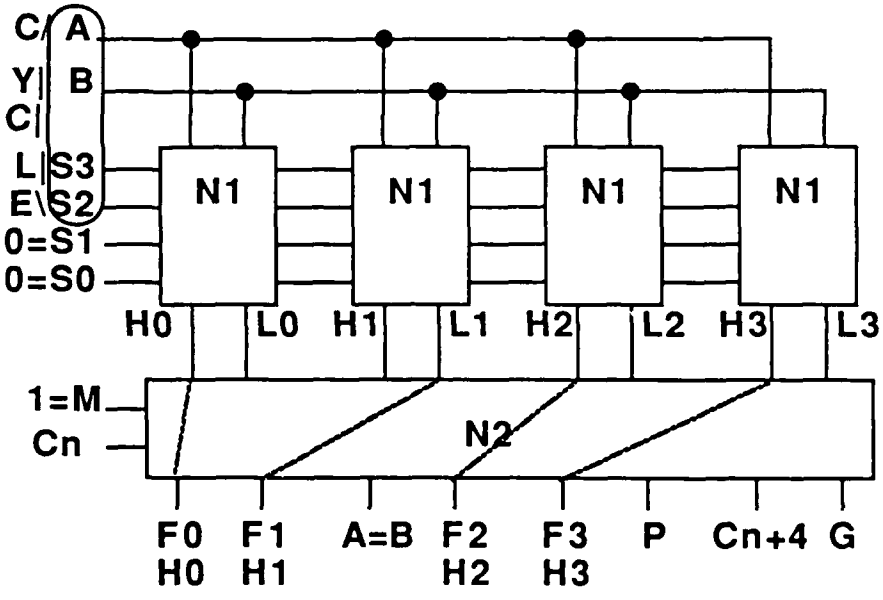


2^{14} Exhaustive Test Patterns

PET VERIFICATION OF LI FUNCTIONS
16 TEST INPUT PATTERNS

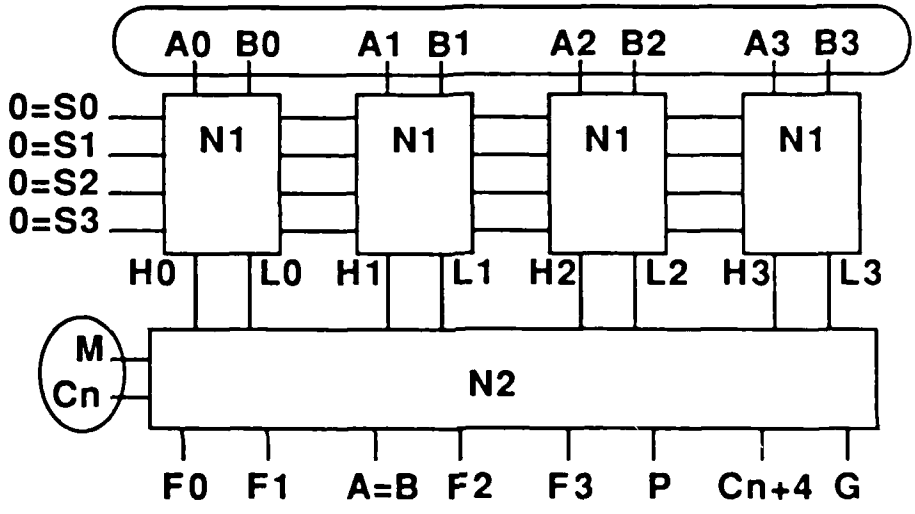


PET VERIFICATION OF HI FUNCTIONS
16 TEST INPUT PATTERNS



PET VERIFICATION OF N2 FUNCTIONS
Hi = (Ai Bi)', Li = Ai' (No HiLi = 01)

3^4 AB tests 2^2 MCn tests, = 324 tests



PET SEGMENT FUNCTION VERIFICATION

181 ALU Network Test Lengths	
Exhaustive Test	4096
Output Function Verification	4096
Segment Function Verification	356

EXHAUSTIVE and PSEUDO-EXHAUSTIVE TEST

Is 2^n Test Length Short Enough?
YES – Use Exhaustive Test

NO – Compute Dependency Matrix
and Partitioned Dependency Matrix

Is U(p, w) Test Length Short Enough?
YES – Use PET Output Verification

NO – Determine Network Segments
for Segment Function Verification

Place Multiplexers or Compute
Inputs to Sensitize Outputs

EXHAUSTIVE and PSEUDO-EXHAUSTIVE TEST

EVALUATION

Possibly Required

Fault-Free Simulation
Dependence Matrix Calculation
Partitioning of D(N)
Segmentation Calculation
Sentization Calculation

EXHAUSTIVE and PSEUDO-EXHAUSTIVE TEST

EVALUATION

Not Required

Fault Simulation

ATPB (LASAR) program
Testability Measure Program

Circuit Modification to increase fault coverage

Very high fault coverage:

Single Stuck Faults All Detected

END
DATE
FILMED
4-88
DTIC